

---

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA  
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,  
TÎRGU-MUREŞ  
PROGRAMUL DE STUDII ...**

**TITLUL PROIECTULUI DE  
DIPLOMĂ**

**PROIECT DE DIPLOMĂ**

**Coordonator științific:  
Ş.I.dr.ing. Turos László-Zsolt**

**Absolvent:  
Lukács Botond**

**2022**

UNIVERSITATEA "SAPIENTIA" din CLUJ-NAPOCA  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  
Specializarea: ...

**Viza facultății:**

**LUCRARE DE DIPLOMĂ**

Coordonator științific:

Candidat:  
Anul absolvirii:

**a) Tema lucrării de licență:**

**b) Problemele principale tratate:**

**c) Desene obligatorii:**

**d) Softuri obligatorii:**

**e) Bibliografia recomandată:**

**f) Termene obligatorii de consultații:** săptămânal

**g) Locul și durata practicii:** Universitatea Sapientia,  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

**Primit tema la data de:**

**Termen de predare:**

**Semnătura Director Departament**

**Semnătura coordonatorului**

**Semnătura responsabilului  
programului de studiu**

**Semnătura candidatului**

---

## **Declarație**

Subsemnatul/a **Lukács Botond**, absolvent al specializării **Calculatoare**, promovația 2022 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Târgu Mureș,

Data:

## **Extras**

Extract

**Cuvinte cheie:**

**SAPIENTIA ERDÉLYI MAGYAR  
TUDOMÁNYEGYETEM  
MAROSVÁSÁRHELYI KAR  
SZÁMÍTÁSTECHNIKA SZAK**

**Elektronikai alkatrész teszter**

**DIPLOMADOLGOZAT**

**Témavezető:**  
Ş.l.dr.ing. Turos László-Zsolt

**Végzős hallgató:**  
Lukács Botond

**2022**

# Kivonat

Napjainkban a mikrovezérlős rendszerekben sok mindenben megtalálhatóak és manapság nagy számítási kapacitással rendelkeznek, általánosan alkalmazhatóak sokféle különböző alkalmazásban. Rengeteg funkciójuk van, az egyszerű LED villogtatástól kezdve komplex rendszerek automatizálásáig. Ezen kívül könnyű külső kiegészítő tartozékokat amelyekkel sokkal szélesebb körben használhatóak.

Ennek hatására a dolgozat célkitűzése egy olyan rendszer kialakítása, amely képes meghatározni egyszerű elektronikai komponenseket és azok megközelítő értékét és ezen kívül a lábkiosztását is amennyiben ez szükséges. Az elektronikában sok féle egyszerű komponenssel találkozhatunk, mint ellenállások, kondenzátorok, tranzisztorok. Viszont egy áramkör építésénél jó tudni, hogy az a komponens pontosan mi, ez legfőképpen igaz a különböző félvezetőkre. Sok esetben az azonosítója lekopott, vagy nem található adatlap így nehéz beazonosítani, hogy pontosan mi az a komponens. Erre szolgál az „elektronikai alkatrész teszter” amely automatikusan meghatározza, vagy kiírja, hogy hibás alkatrész ha nem ismeri fel vagy sérült az tesztelt alkatrész. A rendszer egy mikrovezérlőt és egy kijelzőt használ az komponens azonosítására és arról levő adatok kijelzésére a felhasználó felé. Az azonosítás teljesen automata, csupán csatlakoztatni kell az ismeretlen komponenst és egy gombot megnyomni.

A dolgozatban a mikrovezérlős alkalmazásokról és azok tervezéséről, alkatrészek felismeréséről és méréséről lesz szó.

**Kulcsszavak:** mikorvezérlő

# **Abstract**

Abstract

**Keywords:**

# Tartalomjegyzék

<b>1. Bevezető</b>	<b>1</b>
1.1. Téma meghatározása . . . . .	2
<b>2. Elméleti megalapozás és szakirodalni tanulmány</b>	<b>3</b>
2.1. Elméleti alapok . . . . .	4
2.1.1. Mikrovezérlők . . . . .	4
2.1.2. Analóg Digitál Konverter . . . . .	5
2.1.3. Digitál Analóg Konverter . . . . .	6
2.2. Felhasznált eszközök . . . . .	7
2.2.1. Raspberry pi pico . . . . .	7
2.2.2. ILI9341 kijelző . . . . .	8
2.2.3. Digitál Analóg Konverter . . . . .	9
2.2.4. Analóg kapcsoló . . . . .	10
2.3. Felhasznált Technológiák . . . . .	11
2.3.1. SPI . . . . .	11
2.4. Hasonló eszközök . . . . .	11
<b>3. Rendszer specifikációja és architektúrája</b>	<b>13</b>
3.1. Rendszer Követelmények . . . . .	13
3.1.1. Funkcionális követelmények . . . . .	13
3.1.2. Nem funkcionális követelmények . . . . .	14
3.1.3. A programozási nyelv . . . . .	14

3.1.4. Mikrovezérlő felprogramozása . . . . .	15
3.2. A rendszer Blokk váza . . . . .	15
3.3. A rendszer osztály diagrammjai . . . . .	16
3.3.1. Kijelző vezérlő . . . . .	16
3.3.2. Mérő áramkör osztály diagramma . . . . .	16
3.3.3. Számításokért felelő osztályok . . . . .	17
3.4. A mérés folyamat ábrája . . . . .	18
3.5. Mérő áramkör . . . . .	19
3.6. A mérés menete a felhasználó által . . . . .	21
<b>4. Részletes tervezés</b>	<b>22</b>
4.1. Szoftver rész . . . . .	22
4.1.1. A beépített ADC használata . . . . .	22
4.1.2. Digitál analóg Konverter . . . . .	23
4.1.3. Beépített SPI használata . . . . .	24
4.1.4. Kijelző vezérlése . . . . .	24
4.1.5. Analóg kapcsoló . . . . .	26
4.2. Hardver rész . . . . .	27
4.2.1. Problémák az áramkör tervezésekor . . . . .	28
4.2.2. Áramkör elkészítése . . . . .	29
4.3. Szimulációk . . . . .	30
<b>5. Üzembe helyezés és kísérleti eredmények</b>	<b>32</b>
<b>6. A rendszer felhasználása</b>	<b>33</b>
<b>7. Következtetések</b>	<b>34</b>
<b>Irodalomjegyzék</b>	<b>34</b>
<b>A. Függelék</b>	<b>36</b>

# Ábrák jegyzéke

2.1.	ADC ofszet hibája . . . . .	6
2.2.	Pico pinout . . . . .	8
2.3.	ILI9341 kijelző . . . . .	9
2.4.	Analóg kapcsoló funkcionális táblázata . . . . .	10
2.5.	Eredeti teszter bekötési rajza . . . . .	12
3.1.	A rendszer blokk váza . . . . .	16
3.2.	A kijelző osztály diagramja . . . . .	17
3.3.	A vezérlő osztályok diagramja . . . . .	17
3.4.	A számításokat elvégző osztályok diagramja . . . . .	18
3.5.	A mérés fázisai . . . . .	19
3.6.	Saját teszter bekötési rajza . . . . .	20
4.1.	Kapcsolási táblázat . . . . .	26
4.2.	NYÁK első verziója . . . . .	28
4.3.	NYÁK végső verziója . . . . .	29
4.4.	A kész áramkör . . . . .	30

# **1. fejezet**

## **Bevezető**

A mikrovezérlős rendszerek manapság az életünk minden részében megtalálhatóak, kis méretük, alacsony áruk és meglehetősen nagy teljesítményükkel sok mindenre általánosan használhatóak. Ez nagyban csökkenti a tervezési költségeket, mivel nem kell egy specifikus logikai áramkört kialakítani minden egyes alkalmazási területre, csupán új program kódot kell feltölteni és használható egy teljesen más cébra.

A mikrovezérlő könnyen összeköthetőek külső eszközökkel amellyel rengeteg minden meg lehet valósítani és bővíteni a lehetőség és szabadon vezérelhetők a GPIO-n (General Purpose Input Output) keresztül sok minden el lehet érni, az egyszerű LED kapcsolatától komplex jelek generálásáig. Általános esetben a mikrovezérlők csak a legfontosabb részeket tartalmazzák, mint az Analog Digital Converter amivel egy analóg jelet alakít egy digitális jellé amit a processzor fel tud majd dolgozni, ez legfőképpen azért van, mert nem mindenkinél van szüksége mindenre így akinek szüksége van az egyszerűen az külsőleg csatolja hozzá.

Az összeállított rendszert szabadon lehet vezélni így automatizálni lehet vele folyamatokat, mivel egyszerű megismételhető teszteket végezni velük, miközben képesek valós időben mérni a rendszer viselkedését. Ennek a feladatnak is ez a lényege, az alkatrészek tesztelése egy gyors és automatizált módon.

## 1.1. Téma meghatározása

A dolgozat célja egy olyan eszköz tervezése, amit bárki elektronikai ismeret nélkül is egyszerűen használni lehet. Sok esetben a feliratok az alkatrészeken nehezen látható, lekopott vagy egyszerűen nincs feltüntetve. Ilyen esetben sok segítséget tud nyújtani egy olyan eszköz ami gyorsan meg tudja határozni a komponenst és a lágkiosztását is amennyiben ez fontos.

Ez különösen nagy segítséget nyújt kezdőknek akik még kevésbé ismerik az alkatrésekét és az adatlapjai meg nagyok és komplexek számukra és a fő információk megjelenítése néhány sorban. Haladóknak is nagy segítség, mivel az ellenállást színkódjátról egyszerű meghatározni, viszont a teszterrel meg lehet határozni, hogy az alkatrész hibás-e, vagyis ha a tranzisztor kiégett akkor az is letesztelethető.

A teszternek 3 teszt terminálja van, ebbe kell az ismeretlen komponenst bekötni és képes egyszerű elektronikai komponensek (ellenállás, dióda, tranzisztorok, stb.) automatikus felismerését és az adatainak meghatározására. Viszont nem képes bonyolultabb áramkörök azonosítására aminek összesen több mint 3 lába van.

Megvalósítás során a költégek csökkentése a cél, miközben a pontosság nem csökken nagyban. Két verzió is összeállítható, az egyik egyszerű ellenállásokkal és a második egy DAC (Digital Analog Converter) segítségével. Mindkettő alkalmas a komponensek meghatározására, viszont az ellenállásos verzió nem alkalmas karakterisztika diagram kirajzolására, viszont sokkal olcsóbb, mivel nem használ egy külső DAC-ot.

Mérés eredménye kikerül egy kis kijelzőre és grafikus felületen is megtekinthető amennyiben egy számítógéphez van csatolva. Viszont a 2 közül legalább az egyikre szükség van, különben a mérés eredménye nem lesz látható. A kijelzőt nem kötelező alkalmazni, viszont annélkül csak egy laptop/-számítógéphez kapcsolva lehet használni.

Ehhez szükséges egy processzor, viszont manapság a mikrovezérlők nagy számítási kapacitással rendelkeznek meglehetősen alacsony áron és néhány ellemállásból és vezetékből otthon is összeállítható.

Karakterisztika diagramm kirajzolása is fontos, viszont ez leginkább a tranzisztoroknál fontos.

## **2. fejezet**

# **Elméleti megalapozás és szakirodalni tanulmány**

Első lépésként tanulmányoztam egy hasonló terméket, amely hasonlóan működik, viszont kevesebb funkcionalitással. Ebből megismerve a működési elvét és ezt fejlesztve terveztem meg a teszteremet. Legfontosabb része a teszternek egy mikrovezérlő, amely a rendszer magját adja, erre a célra egy Raspberry Pi Pico-t választottam, a nagyszámú GPIO-ja miatt, nagy teljesítménye miatt és nagy sebességű beépített hardveres kommunikációs protokollokkal (SPI). Ezután egy kijelző következett, amelyeken a teszter ki tudja jelezni az adatokat az adott komponensről.

Erre a célra egy ILI9341 kijelzőt használtam, ez egy 2.2” méretű színes TFT kijelző, és erre íródnak ki az adatok a felhasználó fele ami SPI-n keresztül kommunikál a mikrovezérlővel. Ezen kívül van 2 LED, amely a rendszer státuszát jelzi egyszerű színkódokkal.

Az teszteléshez szükséges áramkört a DAC segítségével történik és a mikrovezérlő csak a feszültség értékeit nézi, erre a célra egy DAC8565 chipet használtam amely szintén SPI-n keresztül kommunikál a mikrovezérlővel.

Ennek 3 kimenete egy-egy 3 kimenetes analóg kapcsolón keresztül különböző ellenállásokra kapcsolódnak a nagyobb precizitás elérése érdekében és az esetleges rövidzárlat esetén is az áramerősségeg biztonságos szinten tartásáért, minden esetben a létrehozott áramkörön lesz egy ellenállás, ami az áramerősséget limitálja, hogy esetlegesen ne tegye tönkre a tesztelés alatt levő alkatrészt.

A mikrovezérlőnek 3 ADC (analóg-digitál konverter) portja közvetlen rá van csatolva egy lábra

ahová majd a tesztelni kívánt komponens kerül. A rendszernek van egy külső referencia feszültsége, ami egy stabil 3.3V-ot biztosít az ADC referenciaként és DAC referenciának.

A pontos részleteket az a következőkben lesznek leírva.

## 2.1. Elméleti alapok

### 2.1.1. Mikrovezérlők

A mikrovezérlők már a elterjedtek a háztartási eszközök körében is. Legfőképpen az általánosságuk miatt, vagyis egyszerű különböző eszközöket vezérelni velük miközben az általánosságukból fakadóan alacsony az áruk.

Amíg régen egyedileg kellett a hardvert felállítani különböző eszközöknek az alapoktól kezdve, ami időigényes és költséges, mivel egy mérnök csapat meg kell tervezze, tesztelje és kivitelezze ami az alacsony darabszám miatt költséges egy darabra nézve. Ennek a hátránya ezen kívül még az, hogy nehezen módosítható és még hasonló eszközök eszközök közt sem cserélhetőek. Viszont maga a rendszer hatékonyabb lehetett, mintha egy általános rendszerből lett volna kialakítva, ha a működés szempontjából van vizsgálva.

Manapság sokkal jobban megéri tömeggyártani egy mikroprocesszort ami általánosan használható rengeteg különböző eszközben, miközben a rendszerhez csupán hozzá kell csatolni a szenzorokat. Ehhez még nagy segítség a standardizált kommunikációs protokollok is a szenzorokkal, amely segítségével könnyedén összekapcsolhatóak a mikrovezérlővel, legtöbb esetben hardver szinten ismeri ezeket a protokollokat így gyorsan és hatékonyan képes kommunikálni. A leggyakrabban használt protokollok az  $I^2C$  és SPI. Viszont manapság sok mikrovezérlőn a WiFi és Bluetooth is megtalálható amivel vezeték nélkül is össze lehet kötni az eszközöket, ezek az eszközök leginkább az IoT - Internet of Things (Dolgok Internete) alapú rendszerek által használatosak.

Ezek a protokollok legtöbbször egyszerűek, így lehetséges szoftveresen is megvalósítani viszonylag egyszerűen a mikrovezérlő digitális kimeneteit használva. Viszont ezzel az a gond, hogy helyet foglal a memóriában, így nagy projektek esetén memória gondok léphetnek fel. A másik nagy gond, hogy ha időzítés érzékeny a protokoll akkor nehezebb betartani az időzítéseket, különösen ha nagy

frekvenciájú jeleket kell kiküldeni. Egy ilyen példa egy VGA jel generálása, a szinkronizáló jelek előre meghatározott időben és csúszások nélkül kell megérkezzen, miközben pontos időben kell kikerüljön az adat is. Ez nehéz feladat teljes mértékben szoftveres megoldással megvalósítani.

Egyes mikrovezérlők ezért lehetővé tették a programozható be/kimeneteket. Ennek a lényege, hogy néhány kimenetet egyszer felprogramozva a processzortól függetlenül működik, mintha egy hardveres megvalósítás lenne, ezzel meg sokkal egyszerűbb egy nem alapértelmezetten támogatott protokoll megvalósítása, miközben a processzor nincs leterhelve ezzel a feladattal.

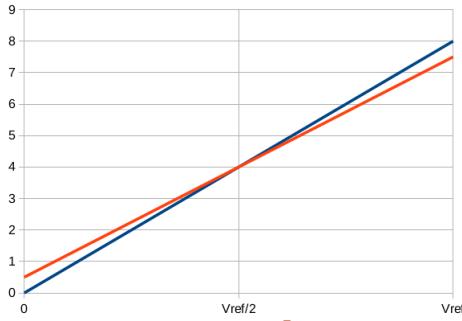
## 2.1.2. Analóg Digitál Konverter

Az Analóg Digitál Konverter (ADC) feladata, hogy egy analóg jelet digitális jellé alakítson át, mivel a processzor csak digitális jeleket képes feldolgozni. Viszont 1 bitben nem használható, mivel szinte az összes adat elveszne, ezért az ADC több biten tárolja el az analóg mérés eredményét. Ez legtöbbször 8-16 bit közti értékek közt található. A maximális érték amit az ADC eredménynek kiad az akkor történik, amikor az analóg jel feszültsége megegyezik az ADC referencia feszültség szintjével és 0 értéket ad ha a jel feszültsége 0V és a két érték között lineáris összefüggés van amely meredeksége 1. Erre több megoldás is létezik, viszont mindenik ugyan azt a célt éri el, megállapítja, hogy melyik 2 érték közé esik a feszültség szint.

Viszont ez a valóságban nem ilyen egyszerű, mivel a komponensek amik az ADC-t felépítik nem tökéletesek így zajok és csúszások jelenhetnek meg. Ilyen hibák lehetnek az ofszet hibák, ebben az esetben a végértékek nem felelnek meg az elvárt értékeknek, ilyen példa a amikor 0V-os feszültség esetén az ADC nem 0-t térít vissza, hanem egy nagyobb számot, míg  $V_{ref}$  esetén egy kisebb értéket.

Az alábbi ábrán egy ilyen eset látható [2.1] , a kék vonal az elvárt érték és a piros az aktuális érték, amit az ADC térit vissza a mérési tartományban. Ebben az esetben csupán ez a hibája az ADC-nek az átláthatóság kedvéért.

Másik jellegzetes hiba a nem linearitás, ennek jellemzője, hogy a mérési tartományon belül még az ofszet hibát is figyelembe véve nem azt az eredményt kapjuk, mint amit elvártunk amelyet nehezebb korrigálni szoftveresen, mivel amíg az ofszet hibát egyszerűen lehet korrigálni csupán a 0 és  $V_{ref}$  feszültség szintek megmérésével, addig ezt a hibát sokkal nehezebb kiküszöbölni, mivel az ADC ered-



2.1. ábra. ADC ofszet hibája

ménye nem lineáris, így több ismert mérési pontot kell felvenni és a számítás is sokkal bonyolultabbá válik. Ezen kívül lehetnek kvantálási hibák is, amikor az ADC egy bizonyos értéket sosem ad eredménynek, hanem a nála egyel kisebb/nagyobb értéket. Ez nem okoz nagy gondot azoknál az ADC-nél ahol nagy a felbontás, viszont az alacsony felbontású eszközök esetén nagyobb hibát eredményez.

### 2.1.3. Digitál Analóg Konverter

A Digitál Analóg Konverter feladata, hogy a processzor által értelmezhető digitális értékeket analóg jelekké konvertálja amik a külső perifériáknak szükségesek. Gyakran használják ahol pontos analóg jeleket kell vezérelni, ilyen egy VGA jel vagy zene lejátszása, viszont bármire felhasználhatóak ahol az egyszerű digitális jelek nem elegendők. Ebben a projektben az alkatrészek azonosítására és a karakterisztika diagram kirajzolására van felhasználva.

Működése hasonlóképpen működik, mind az Analóg Digitál Konverter-é, viszont fordítva. Egyes DAC-eknek van belső referencia feszültségük, vagy kell külsőleg egyet alkalmazni. Ez a jel lesz DAC maximális analóg kimenete amikor a digitális bemeneten a maximális érték érkezik és 0V lesz a kimenete amikor 0 értékű digitális érték érkezik. Ideális esetben e két pont között az átmenet lineáris, így minden köztes értékre egy lineáris összefüggés alapján ki lehetne számolni [2.1], ahol D a digitális bemenet ami 0 és  $2^{res}$  között kell legyen.

$$V_{out} = V_{ref} \frac{D}{2^{res}} \quad (2.1)$$

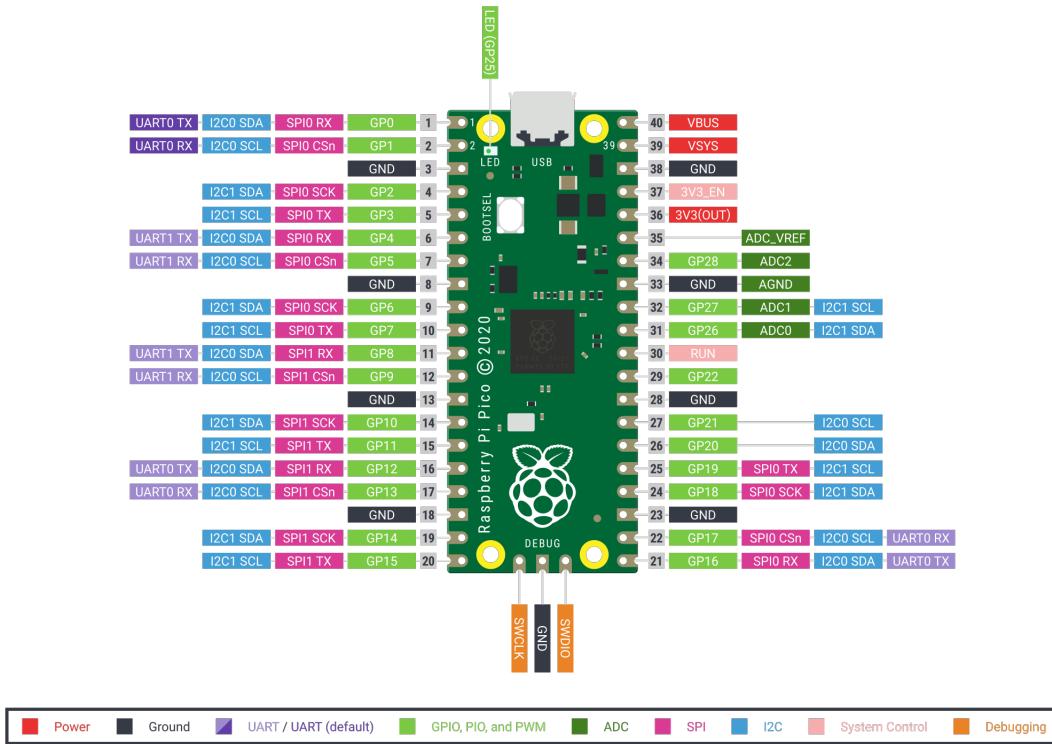
## 2.2. Felhasznált eszközök

### 2.2.1. Raspberry pi pico

Ez a teszter legfontosabb eleme [2.2], ez a vezérelti a teszter többi részét. A Raspberry által kifejlesztve, viszont nem, mint a többi termékük ezen nem fut egy operációs rendszer és használható, mint egy számítógép, hanem mint egy hagyományos mikrovezérlő. A mikrovezérlő magját egy Dual-core ARM Cortex M0+ processzor, aminek a frekvenciáját dinamikusan változtatható 16Mhz és akár 280Mhz között. A memóriája is elegendő 264KB of SRAM a változók tárolására a program futása során és 2MB Flash memória a program kód tárolására. Lehetőség van csak RAM-ból futtatni is a kódot, ilyenkor nagyobb órajel is elérhető ami elérheti a 450Mhz-et is.

A vezérlő úgy van megtervezve, hogy THC (Through Hole Component) és SMT (Surface Mount Component) komponensként is lehet alkalmazni, ezt a castellated pinekel éri el, amelyeket lehet forrasztani közvetlen SMT komponensként és a pinek távolsága annyi, mint egy átlagos breadboard sor-távja így egy male-male header forrasztásával breadboardon is alkalmazható.

Hardveresen megtalálható 2xSPI, 2xI2C, 2xUART, 3x12-bit ADC, 16xszabályozható PWM csatorna és összesen 26 GPIO van kivezetve a felhasználó fele. Egy pin több dologra is képes ez az alábbi pinout diagramon is látható. Programozás során beállítható, hogy USB-t használja soros portként így egyszerűen használható egy számítógéppel való kommunikációra. Megtalálható 8 PIO (programable I/O) amivel hardveres protokollokat lehet létrehozni, ha valami egyedi kell egy különleges komponens vezérlésére.



2.2. ábra. Pico pinout

## 2.2.2. ILI9341 kijelző

Az adatok kijelzésére egy ILI9341 [2.3] kijelzőt használtam. Egy egyszerűbb kijelző is megfelelne a célra, akár csupán egy karaktereket megjelenítő kijelző is elegendő lenne a célra, ha nem kellene a karakterisztika diagram. A célra megfelelne egy egyszerű OLED kijelző is. Viszont ez a kijelző már a rendelkezésre állt a projekt tervezése előtt is, így fel lehetett használni a kijelzőt erre a projektre.

Ezen fognak megjelenni az mérés eredményei, mint a komponens értékei, a lábkiosztása és a karakterisztika diagramja. A képernyő képes 16 bites színes kép megjelenítésére, viszont ez nem fontos ennél a projektnél. Viszont a nagy 240\*320-as felbontása és a 2.2" kijelző mérete nagyban segít, hogy könnyen olvasható legyen. A kijelző rendelkezik beépített háttérvilágítással is, így fényes helyen is jól látható. A kijelző SPI protokollon keresztül kommunikál a mikrovezérlővel és rendkívül stabil, akár 68Mhz-es órajellel is stabilan és megbízhatóan megkapja az adatokat a mikrovezérlőtől. Ennek az eredménye a rendkívül gyors képernyő váltás, így a felhasználó nem kell arra várjon, hogy

az adatok megjelenjenek a kijelzőn.

Az adatok beírása pixel szinten történik amelyek a kijelző memóriájában elmentődnek, előre meg kell határozni, hogy milyen zónában történik az írás és azután lehetséges csak az adat küldése és belsőleg automatikusan a következő mezőt címzi meg, ahogy megérkezett az adat az előzőnek. Ennek az eredménye, hogy írás során nem kell figyelembe tartani a kijelző belső memória címét, csak az adat sorrendjét. Ez egyszerűsíti a programozó dolgát és csökkenti a küldési időt.



2.3. ábra. ILI9341 kijelző

### 2.2.3. Digitál Analóg Konverter

Az analóg jelek generálására egy DAC8565IAPWR [1] van alkalmazva, amelynek felbontása 16 bit. Ennek 4 egymástól független kimenete van, amelyek belsőleg pufferelve vannak, így terhelés esetén sem változik.

A DAC-nak van egy belső 2.5V-os feszültség referenciája, viszont ez külsőleg felülírható egy külső feszültség referenciával. Ebben az esetben szükség volt erre, mivel a teljes ADC mérési tartományra szükség van ami 3.3V. Ez egy egyszerű 3.3V-os feszültség referenciával lett elérve, viszont ebben az eseten egy parancsot kell küldeni a DAC-nak, hogy a külső referenciát használja, mivel nem áll át automatikusan, rövid időre nem gond, hogy minden aktív, viszont nem ajánlott huzamosabb ideig ilyen helyzetben tartani.

A DAC SPI protokollt használ amiből 8 bit a parancs bit és 16 bit az adat. Lehetőség van az adatok azonnali betöltésére és a kimenet változtatására, és arra is van lehetőség, hogy csak az adatot elmentse és egyszerre frissítse az összes kimenetet. Erre lehetőség van szoftveresen is, amit az SPI parancs részében lehet kódolni, ebben a projekt esetében is ez van alkalmazva.

#### 2.2.4. Analóg kapcsoló

Mindegyik analóg jelet egyszerre 1 ellenálláson keresztül rá kell kapcsolni a komponensre, viszont az ellenállás váltható kell legyen mérés közben, nagy sebességgel, így a manuális kapcsolás nem lehetséges. Erre a célra egy analóg kapcsoló van alkalmazva amely egy közös bemenetet 3 különböző kimenetre képes kapcsolni, melyeken az ellenállások találhatóak. A kimenet kiválasztása nem használ kommunikációs protokollt, csak a "IN" portokra érkező digitális jelek segítségével történik. minden esetben maximálisan 1 kimenet aktív a többi a nyílt áramkörben van és arra is van lehetőség, hogy a bemenet 1 kimenetre se legyen kapcsolva. A kimenet kiválasztása a következő táblázattal lehetséges.

[2.4]

FUNCTION TABLE					
IN1	IN2	COM TO NO0	COM TO NO1	COM TO NO2	
L	L	OFF	OFF	OFF	
H	L	ON	OFF	OFF	
L	H	OFF	ON	OFF	
H	H	OFF	OFF	ON	

2.4. ábra. Analóg kapcsoló funkcionális táblázata

## 2.3. Felhasznált Technológiák

### 2.3.1. SPI

A kommunikációs protokoll amit a kijelző és a DAC használ. A mikrovezérlőnek vannak beépített függvényei és specifikus pinei amellyel képes hardver szinten kommunikálni külső eszközökkel. Ez egy full duplex protokoll, vagyis képes egyszerre minden eszköz adatot küldeni, anélkül, hogy egy mászt zavarnák. Ez 2 külön vezetéken keresztül történik, órajel szinkronizációval.

Az SPI egy mester szolga alapú protokoll, általános esetben 1 mester van (pl. egy vezérlő), de lehet több mester is, amennyiben csak 1 mester aktív és a többi mester a kimeneteit magas impedanciára kell helyezze (High Z), hogy ne befolyásolja az adatok küldését.

A szolgák száma bármennyi lehet, viszont nem szokott nagyszámú lenni, mivel minden szolgának egy külön vezeték kell az aktiválásra, ez a CS (chip select), amely legtöbb esetben aktív és csak akkor lehet a bizonyos eszközzel kommunikálni, amennyiben ez első lépéssben megtörtént.

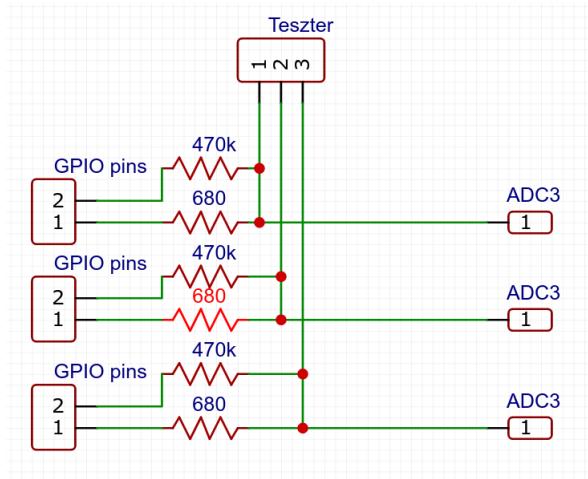
Az SPI egy órajellel szinkronizált protokoll, az órajelet a mester szolgáltatja, amely nincs standardizálva, csupán az eszköz maximális órajellel frekvenciája alatt kell legyen. Az eszközöktől függően nyugalmi helyzetben az órajel lehet alacsony vagy magas szinten.

Az adatok küldésére 2 vezeték szükséges, ezek a MOJSI (master out slave in), ezen a vezetéken érkezik az adat a mester felől a szolga felé. Az adatok kiolvasása nincs standardizálva, így az eszköztől függően az adatot olvashatja az órajel felfutó vagy a lefutó élén. Az adat a mester felé a MISO (master in slave out) vezetéken érkezik, viszont a mester kell órajelet generáljon, hogy a szolga el tudja küldeni az adatokat.

## 2.4. Hasonló eszközök

Egy hasonló rendszer már megvalósult [2] ami egy Arduino [3] mikroprocesszoron alapul. Ez egy egyszerűbb rendszer, ami csupán ellenállásokat használ az alkatrészek felismerésére és egy kijelzőt az adatok megjelenítésére. Ebből több fejlesztés is kialakult, több minden tesztelésre és nagyobb pontosság elérésére, miközben a rendszer egyszerűségét fenntartani. Ezek a rendszerek viszont nem használnak DAC-ot és ezért nem képesek karakterisztika diagramot készíteni. Ezen kívül nem csat-

lakoztatható egyszerűen számítógéphez, csupán újraprogramozás céljából, így minden esetben kell tartalmazzanak egy kijelzőt, ami növeli a költségeket. Az általános működésük hasonló, mint ebben a projektben, viszont itt precízen lehet változtatni a feszültséget, nem csak kapcsolni földre vagy tápfeszültségre. Az bekötése hasonlóképpen történik: lásd [2.5]



2.5. ábra. Eredeti teszter bekötési rajza

Mindegyik GPIO pin lehet csatolva földre, vagy tápfeszültségre, de le is lehet kapcsolva, így nem befolyásolja az áramkör működését. Az ADC pin meg lehet ADC üzemmódban, ilyenkor nem befolyásolja az áramkört, viszont lehet földre, vagy tápfeszültségre is kapcsolni, ilyen esetben port ellenállás nélkül csatolódik az áramkörre.

Viszont szintén hordozható egy 9V-os elem segítségével és az eredmények megjelennek egy kis LCD kijelzőn, ebből több verzió is létezik, van amely csak egy karakter kijelzőt használ, van amelyik egy színes kép kirajzolására is alkalmas kijelzőt alkalmaz.

A tesztelés néhány másodpercbe telik, nagy méretű kondenzátorok esetén telhet több időbe, viszont ebben az esetben csak annyi idő, míg a legkisebb ellenálláson keresztül képes feltölteni a kondenzátort.

## **3. fejezet**

# **Rendszer specifikációja és architektúrája**

### **3.1. Rendszer Követelmények**

#### **3.1.1. Funkcionális követelmények**

A rendszer elvár egy 2 vagy 3 lábbal renddelkező alkatsész csatlakoztatását a teszter sockethez, olyan módon, hogy elektronikusan is csatolva legyen. Az alkatrész lehet ellenállás, kondenzátor, dióda vagy tranzisztor, viszont a rendszer csak maximális feszültsége 3.3V, így amennyiben nagyobb nyitó feszültség szükséges az alkatrészenél akkor abban az esetben nem képes detektálni azt.

A rendszer fő követelménye a tesztelés leegyszerősítése, így egy átlagos, elektronikai imeretekkel nem rendelkező felhasználó is egy rövid 2 perces bemutató után is magabiztosan tudja használni. Ezt a felháló által élérhető bemenetek csökkentésével lett elérve, mivel a felhasználó könnyebben tud egy olyan eszközt használni ahol a bemenetek száma minimális.

A tesztelés az indulástól kezdve automatikus, így tesztelés során a felhasználó nem kell semmit se tegyen, a tesztelés vége a 2 jelző LED felkapcsolósával van jelezve és az eredmények a rendszeren megtalálható kijelzőn jelennek meg és amennyiben a rendszer egy számítógéphez van csatolva abban az esetben a Serial porton keresztül itt is megjelennek az eredmények.

### **3.1.2. Nem funkcionális követelmények**

A rendszer lehetőleg nagy gyorsaságú legyen, mivel ha sok időt venne igénybe a mérés akkor abban az esetben gyorsabb lenne leolvasni az alkatrész azonosítóját és az interneten rákeresni, vagy kézzel lemérni egy multiméter segítségével. A mérés átlagosan 3 másodperc alatt el kell végezzen egy teljes azonosítást. Ez a legtöbb esetben rövidebb idő, csupán a nagy méretű kondenzátorok esetében lehetséges ennek túllépése.

A rendszer egyedülálló módon is alkalmazható akár egy külső akkumulátorról is táplálva, így nincs operációs rendszer követelménye az alapszintű működéshez.

Hordozhatóság is elengedhetetlen, hogy könnyedén minden kéznél legyen. Ezért esett a választás az USB-n keresztül való táplálásra, mivel ez manapság sokkal elterjedtebb, mint az eredeti tranzisztor teszter ami egy 9V-os elemet használ ami kevésbé gyakori és elem lévén le tud merülni, pont akkor amikor a legjobban kellene.

A rendszer tervezésekor, mivel ez egy mikrovezérlőn fut így a tárhely és memória limitált. Amiből a legfontosabb a memória, ebből csupán 264KB található, így figyelembe kellett ezt tartani. Itt legfőképpen a kijelzőn volt szükség az optimalizációra, mivel lehetséges lett volna, hogy a nagy része csak a kijelzővel lett volna elfoglalva.

A mérés során a pontosság nem a legfőbb kritérium, viszont egy megközelítő értéket kell adjon. Az értékek +-10 százalékban eltérhetnek a valós értéktől, ami még mindig elégséges az azonosításra.

A projekt verziókövető rendszer jelenlét mellett készült, amivel vissza lehetett állítani esetlegesen elrontott fájlokat működő állapotába és egy biztonsági másolatkéni is szolgált.

### **3.1.3. A programozási nyelv**

A mikrovezérlő programozásához a C++ nyelv van használva. A mikrovezérlő alkalmazni tudja a standard könyvtárakat, és az objektum orientáltságából fakadóan lehet az OOP sémákat alkalmazni, amellyel bonyolultabb feladatokat is meg lehet valósítani már megoldott példák alapján, ezzel lecsökkenve a programozási időt. Mivel a standard C++ legtöbb függvényét ezért a kód hasonló, mintha az egy számítógépre lett volna írva, csupán az alacsony szinten kell egyedi függvényeket meghívni. Ezek a függvények jól dokumentáltak és a legtöbb esetben van egy teljesen működőképes példa program.

A mikrovezérlő csak C/C++ és micropython nyelveken programozható, viszont ennek kevesebb a dokumentációja és lassabb, így nem alkalmas erre a feladatra.

### 3.1.4. Mikrovezérlő felprogramozása

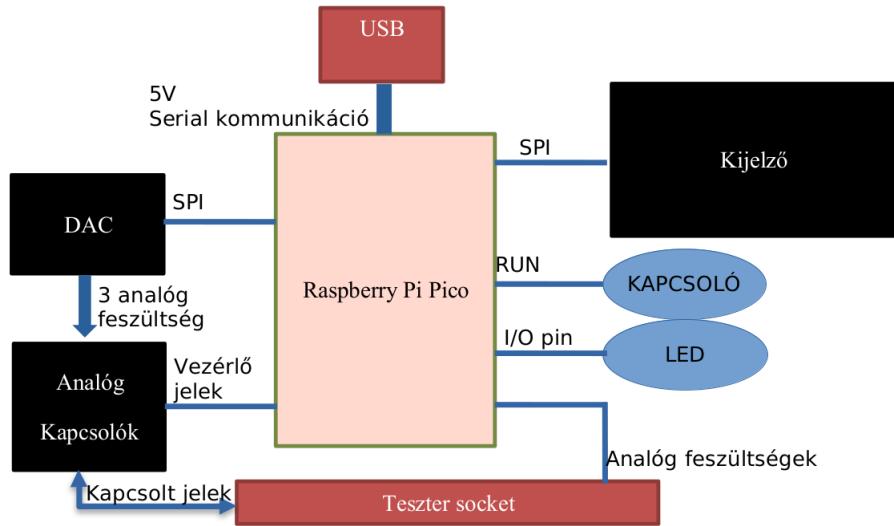
A projekt lefordítására szükséges egy egyedi compiler, ami a programot lefordítsa a mikrovezérlő által megérthető nyelvre. Ez telepíthető a Raspberry oldalán megadott utasításokkal, viszont ez lehetséges, hogy frissül, így telepítés előtt ajánlott azt követni. A compiler Linux és Windows rendszereken is használható. A fordítás után generál egy .uf2 fájlt, ezt kell feltölteni a mikrovezérlőre amit elment a Flash memóriájában és automatikusan indul amit bekapcsol.

A programozásra csak egy micro-USB kábel szükséges. A számítógéphez csatlakoztatás előtt a mikrovezérlőn található BOOTSEL gombot lenyomva kell tartani és így csatlakoztatni a számítógéphez, ekkor megjelenik mint egy külső tároló. Ebbe a tárolóba fel kell másolni a generált .uf2 fájlt, amint ez megtörtént lecsatlakozik és futtatni kezdi a programot.

## 3.2. A rendszer Blokk váza

Az alábbi árbán [3.1] látható a rendszer blokk Diagramja, a fő vezérlő jelekkel. A rendszer 2 SPI perifériával kommunikál, ezek a kijelző és a DAC, az analóg kapcsoló vezérlő jelei egyszerű digitális jelek, mint ahogyan a LED vezérlő jelei. A teszter socket mindhárom lába egy-egy ADC csatornához van csatlakoztatva, ezen méri a mikrovezérlő a tesztelés során a feszültség szinteket. A kapcsoló a RUN bemenetre van kapcsolva, ez engedélyezi a mikrovezérlő futását amíg feszültség érték található rajta vagy nincs bekötve. Amennyiben ez földre kerül akkor a mikrovezérlő leáll és restellődik.

A rendszerhez működéséhez szükséges egy 5V-os feszültség forrás, ez lehet egy általános USB tápforrás, vagy lehetséges direkt 5V csatlakoztatása is. Az áramerősség alacsony, így nem közelíti meg az 500mA-es áramerősség határt amit egy átlagos USB képes leadni. Külső akkumulátorról is táplálható. Amennyiben egy számítógéphez, vagy egy olyan eszközhöz van csatolva ami képes a serial portot olvasni akkor a mérési eredményeket automatikusan elküldi azon keresztül a másik eszköz felé.



3.1. ábra. A rendszer blokk váza

### 3.3. A rendszer osztály diagrammja

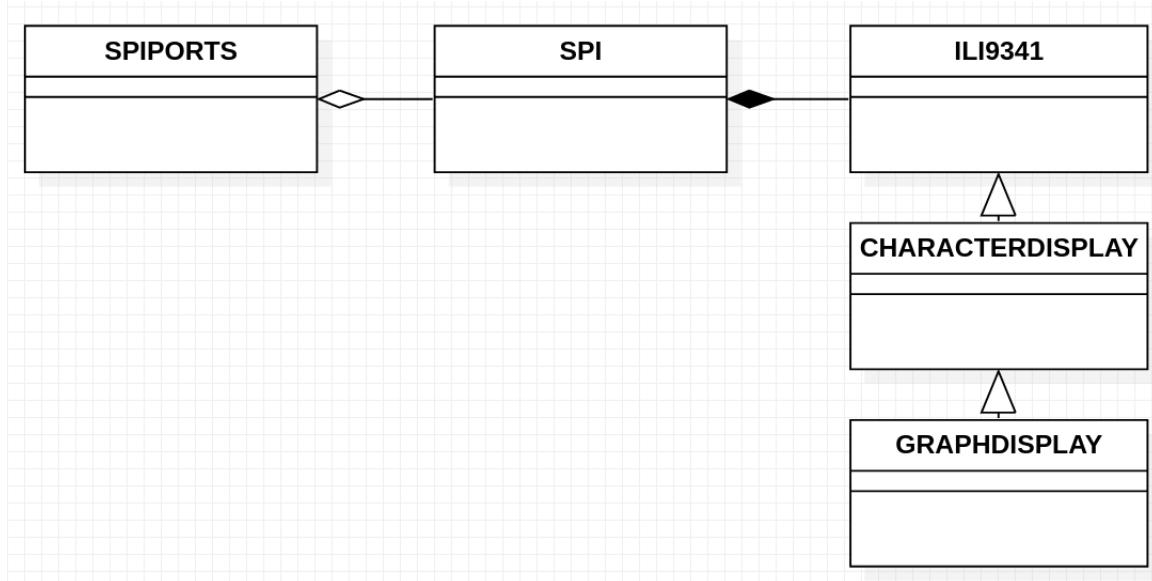
#### 3.3.1. Kijelző vezérlő

Az alábbi ábrán[3.2] a kijelző vezérlésért felelős osztályok láthatóak. Ez a rendszer többi részétől elszigetelt rész, amely csupán megkapja az eredményeket a mérések után és megjeleníti azt.

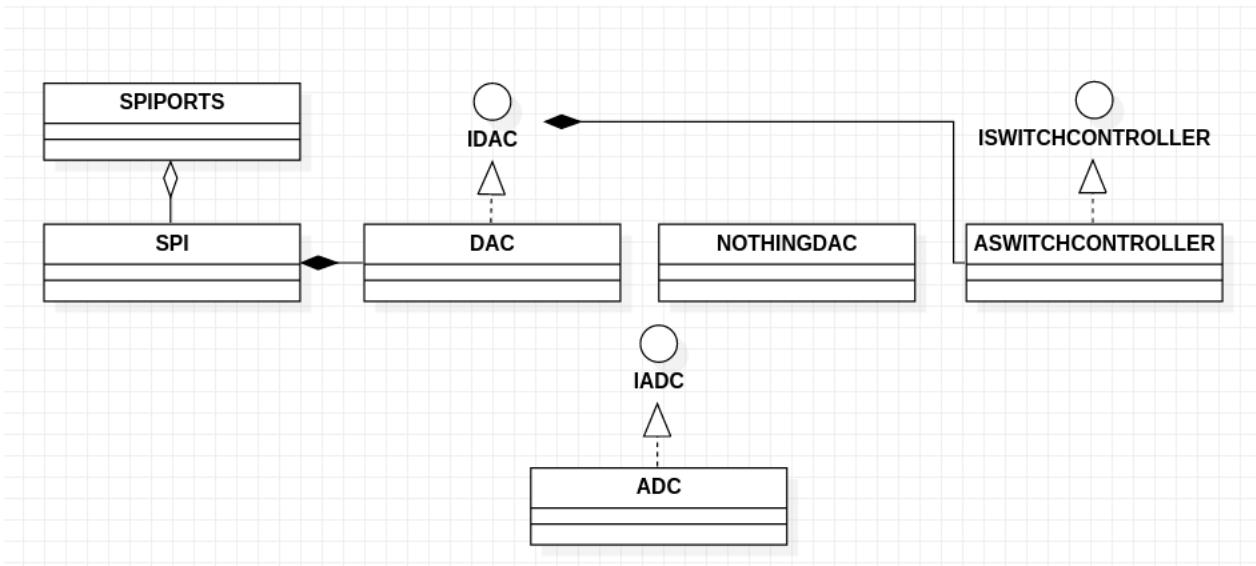
Az ILI9341 osztály az alapszintű vezérlésért felelős, itt csak az adat küldési funkcionalitás és a képernyő inicializására van implementálva. A CHARACTERDISPLAY osztály kibővít ezt és lehetővé teszi egyszerű szöveg megjelenítését a kijelzőn. A GRAPHDISPLAY kibővítve ezt képes megjeleníteni grafikont a kijelzőre beépített skálázással, hogy lehetőleg az egész kijelzőt kihasználja.

#### 3.3.2. Mérő áramkör osztály diagramma

Alábbiakban [3.3] a rendszer az adatok méréséért felelős osztályai láthatóak, ezek az osztályok alkotják a mérő eszközök vezérlését, mint a DAC és az analóg kapcsolók és az adatok mérését. A rendszerben az ADC globális osztály és abból csak egy van.



3.2. ábra. A kijelző osztály diagramja

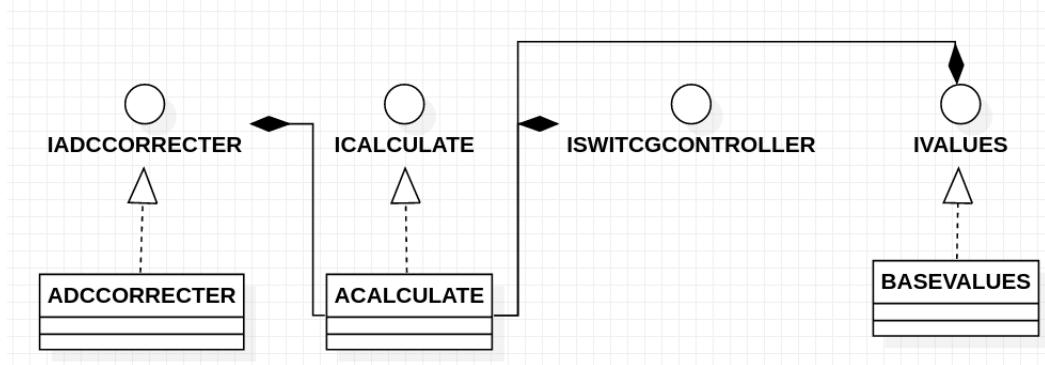


3.3. ábra. A vezérlő osztályok diagramja

### 3.3.3. Számításokért felelő osztályok

A következő osztályok[3.4] felelnek a részeredmények kiszámításáért. A számításokat az ACAL-CULATE osztály végzi, az ADCCORRECTOR osztály feladata az ADC konstans hibájának a ki-

küszöbölésére. Az ISWITCHCONTROLLER az előző [3.3] ábrán látható vezérlő rendszerrel van kapcsolatban. A BASEVALUES osztály a mérés során történt ideiglenes adatok tárolására szolgál.



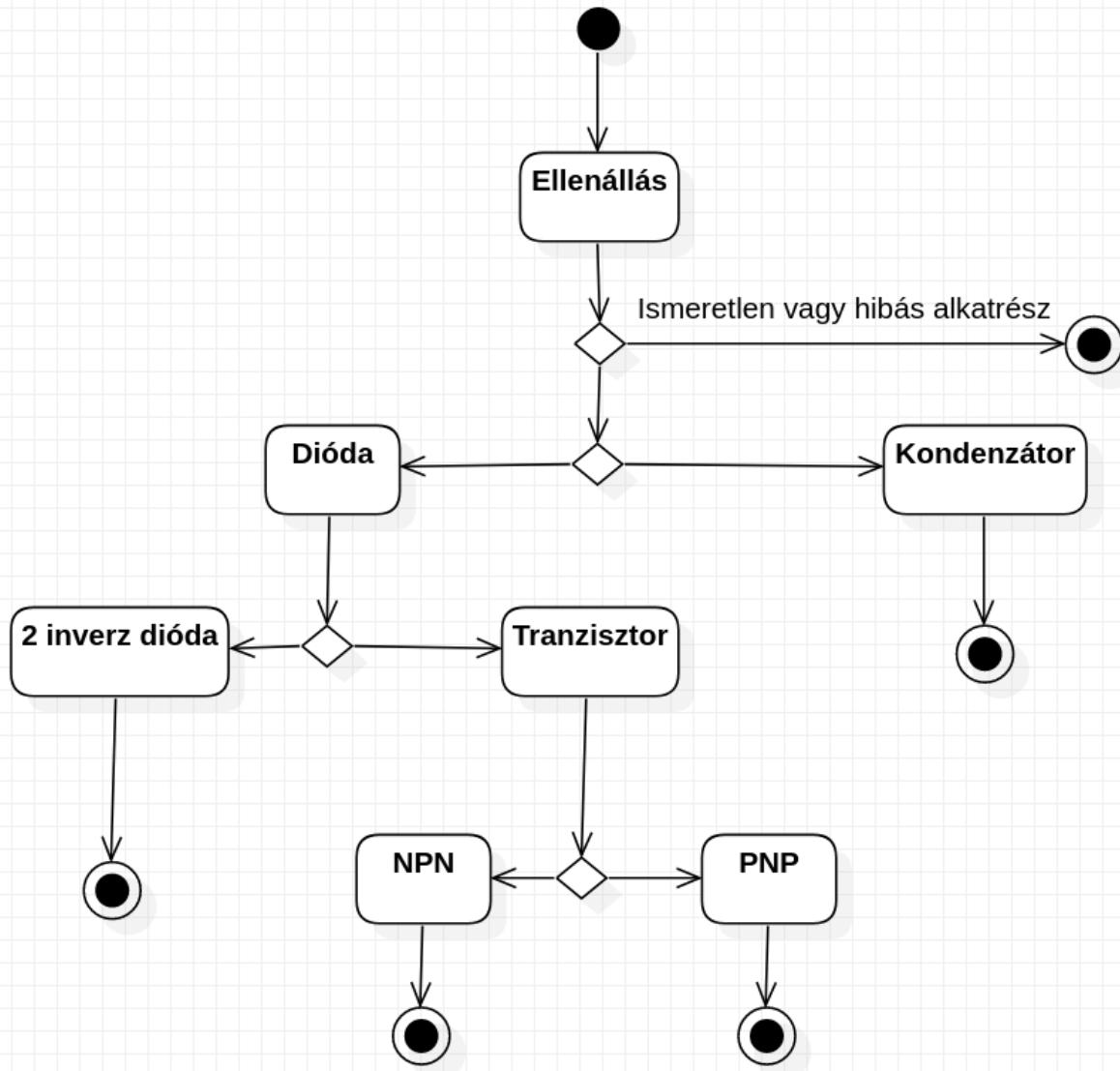
3.4. ábra. A számításokat elvégző osztályok diagramja

### 3.4. A mérés folyamat ábrája

A mérés lépései a következőképpen történik[3.5]. Első lépésben ellenállásra tesztel, ez ha nem talál semmit, vagyis egyik teszter láb sincs csatlakoztatva, vagy nem reagál akkor a tesztelt komponens hibás, vagy nincs csatolva. Amennyiben valamilyen eszköz érzékelve van abban az esetben megpróbálja megmérni az ellenállás értékét, miközben figyelve arra, hogy ellenőrizzen lehetséges diódára is. Ezt úgy oldja meg, hogy a dióda ellenállástól eltérően csak egy irányban vezet és a feszültség esés konstans az áramerősségtől független.

Amennyiben dióda van detektálva abban az esetben a dióda útvonalon folyatódik a tesztelés, különben megpróbálja megmérni az alkatrész kapacitását.

A dióda esetben ellenőrzi, hogy az alkatrész az 2 inverz dióda, és megméri a nyitó feszültségét. Amennyiben nem 2 inverz dióda, akkor abban az esetben teszteli tranzisztorra is, ebben az lépésben képes az NPN és PNP tranzisztorokat is detektálni és meghatározza a tranzisztor lábkiosztását.

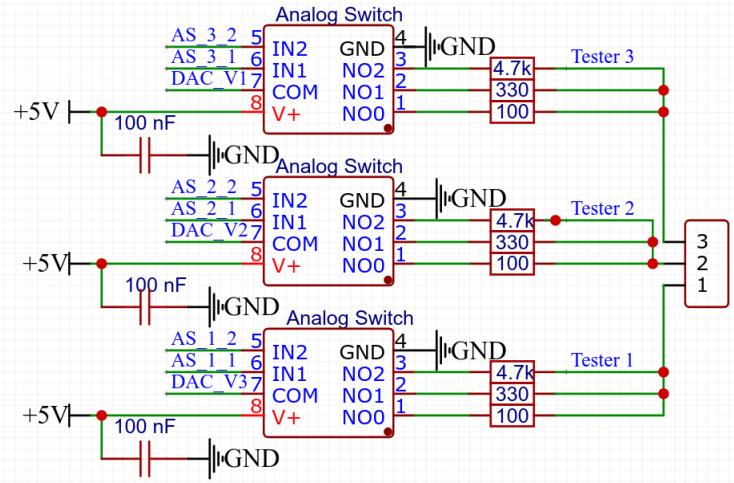


3.5. ábra. A mérés fázisai

### 3.5. Mérő áramkör

A rendszer hasonlóan épül fel, mint a példaként használt rendszer [2], viszont néhány módosítással. Viszont az elméleti alapja hasonló. A komponens lábára egy ellenálláson keresztül feszültséget kapcsolva és a feszültség mérésével az ellenállás után meg lehet állapítani az áramerősséget és a feszültség esést a komponensen. Ezekből az értékekből számítások segítségével ki lehet számolni, hogy mi az

ismeretlen komponens (ennek a leírása és lépései a későbbiekben lesznek részletezve) és értékét. A jelenlegi rendszer mérő áramkör része a következőkben látható. [3.6]



3.6. ábra. Saját teszter bekötési rajza

Ebben az esetben szükséges analóg kapcsolók használata, mivel az eredeti teszter csak digitális jeleket használ amit direkt a GPIO-ról voltak kivezetve, így az ellenállások közti váltásokért belsőleg a mikrovezérlő felt. Viszont ebben az esetben a jelek analóg jellek amit egy külső DAC generál, így a mikrovezérlő nem képes belsőleg kapcsolni ezeket ezért egy 3 kimenetes analóg kapcsoló van alakítmazva erre a célra.

Mivel csak egy kapcsolón csak egy kimenet aktív egy időben, így az ellenállások után a vezetékez összekapcsolhatóak, mivel azok nem fogják befolyásolni a rendszer működését, erre a részre van kapcsolva az ADC egyik csatornája is, mivel itt a feszültség érték azonos a komponens lábán található feszültséggel, így a komponenst nem kell módosítani, hogy a feszültség látható legyen a lábán.

Az analóg jel 0V-3.3V közt lehetséges és a DAC maximálisan 50mA-t képes leadni, viszont mivel a kondenzátorokat ki kell sütni csatlakoztatás előtt és külső tápfeszültség nem kapcsolható így a maximális áramerősség 33mA-nél nem lehet nagyobb semmilyen esetben.

A rendszer átlagolva nézi a feszültséget a komponens lábán, mellyel nagyobb pontosság érhető el, mivel a zajok kevésbé befolyásolják a jelet. Viszont képes időben is mérni a feszültséget, hogy hogyan változik a feszültség ahogy telik az idő. Az ADC 500.000 mérést tud végezni másodpercenként, viszont ez lelassítható egy belső késleltetővel. Ami segítségével a lassan változó jelek is

megfigyelhetőek, mint például egy nagy kondenzátor töltödése.

### **3.6. A mérés menete a felhasználó által**

Az PCB-n található felhasználó által használható eszközök egy kapcsoló és a teszter socket ahová a komponenst kell helyezni. A kapcsoló egy kétállású kapcsoló mely a mikrovezérlőt leállítja, vagy elindítja ami által a mérés automatikusan elindul.

A rendszer tartalmaz 2 LED-et ami a látható a következő fényképen is az áramkörről is a kijelző jobb felső sarka fölött [4.4]. Amennyiben egyik LED sem világít, és a kijelző sem egy fehér hátteret mutat abban az esetben a mikrovezérlő nem kap áramot, ilyenkor ellenőrizni kell az USB kábelt.

Amennyiben a kijelző egy fehér képet mutat, viszont egyik LED sem világít akkor a vezérlő reset állapotban van, ezt a vezérlő alatt levő kapcsoló ON pozícióba való állításával lehet orvosolni.

A mérés kezdetekor a piros LED felkapcsol, majd mérés befejeztekor a Zöld LED is felkapcsol, ebben az esetben a mérés sikeres volt és az eredmény meg kell jelenjen a kijelzőn és az USB-n keresztül elküldve a számítógéphez.

A teszter sockethez nem lehet tápforrást vagy elemet csatlakoztatni, ide tartoznak a nagy méretű kondenzátorok is, a kis méretűeket képes kisütni, de ajánlott csatlakoztatás előtt kisütni ezeket is.

# 4. fejezet

## Részletes tervezés

### 4.1. Szoftver rész

#### 4.1.1. A beépített ADC használata

A mikorvezérlőbe egy 3 csatornás ADC van beépítve, viszont egyszerre csak 1 csatornát lehet olvasni, olvasás előtt át kell váltani arra a csatornára, amelyről az adatot olvasni akarjuk. Az ADC hardver szinten egy 48Mhz-es órajelet kap és egy mérés minimum 96 ciklusba telik, így 500k mérést végez másodpercenként, ezt viszont növelhető dinamikusan szoftveresen, így csökkenthető a mérési gyorsaság.

Az ADC képes DMA-n (direct memory access) [[4]] keresztül kommunikálni a memóriával, így inicializálás után automatikusan elvégzi a méréseket az adott csatornán. A csatorna inicializálása a következőképpen alakul. A méréseket egy előre lefoglalt tömbbe teszi ami jelen esetben a **capture\_buf** tömb, aminek a mérete **CAPTURE\_DEPTH**. A mérés után a tömbben időrendi sorrendben kiolvashatók a mérések.

```
// ezeket csak egyszer a konstruktorban kell meghivni
dma_chan = dma_claim_unused_channel(true);
cfg = dma_channel_get_default_config(dma_chan);

// minden indulás előtt
adc_fifo_setup(
    true, // Write each completed conversion to the sample FIFO
    true, // Enable DMA data request (DREQ)
```

```

    1,      // DREQ (and IRQ) asserted when at least 1 sample present
    true,   // Set sample error bit on error
    false  // Keep full 12 bits of each sample
);
//Set the size of each DMA bus transfer
channel_config_set_transfer_data_size(&cfg, DMA_SIZE_16);
channel_config_set_read_increment(&cfg, false);
channel_config_set_write_increment(&cfg, true);
channel_config_set_dreq(&cfg, DREQ_ADC);
dma_channel_configure(dma_chan, &cfg,
                      capture_buf,           // dst
                      &adc_hw->fifo,        // src
                      CAPTURE_DEPTH,         // transfer count
                      true                   // start immediately
);

// miutan elo van keszitve
adc_run(true); //elinditja a merest
dma_channel_wait_for_finish_blocking(dma_chan); //var mig befelyezi a merest
adc_run(false); //leallitja a merest

```

Az ADC konstans hibával rendelkezik, vagyis 0V feszültséget nem 0-nak érzékel és 3.3V-ot nem 4096-nek (az ADC 12 bites felbontású). Ezért erre a célra egy lineáris hibajavító függvény van alkalmazva. Ennek működési lényege, hogy megméri 0V-on milyen értéket mér az ADC és 3.3V-on. Ezeket az értékeket elmenti és minden jövőbeli mérést a következőképpen korrigál.

```

for (int i = 0; i < samplesSize; i++)
{
    samples[i] = (samples[i] * VCCOffset) - gndOffset;
}

```

## 4.1.2. Digitál analóg Konverter

A DAC SPI-n keresztül kömmunikál, ezért szüksége van egy SPI osztályra amely inicializálja az SPI portokat. A használt portok szintén a **Global.h** fájlból találhatóak meg. A DAC szintén használ az SPI protokokolon kívüli jeleket, viszont ezek a reseten kívül opcionálisak és elégséges lenne közvetlen a földre kötni. Ezekkel a jelekkel lehetséges lenne jelet küldeni, hogy mikor töltse be az adatot a pufferből, viszont ez szoftveresen is megoldható így azt a megoldás van alkalmazva.

Mivel egy külső feszültség referencia van alkalmazva, így a belsőt ki kell kapcsolni a **0x012000** parancs küldésével az SPI-n keresztül. A küldés során 24 bitet vár a DAC amiből az első 8 bit parancsként van értelmezve, míg a hátsó 16bit adatként. mindenik lehetséges parancs fel van sorolva a

**IDAC.h**-ban, amit, így a parancs nevét és a kívánt feszültséggel kell a függvényt meghívni.

### 4.1.3. Beépített SPI használata

A mikrovezérlőnek 2 különálló SPI chatornája van és minden csatornának 2 különböző helyre van kivezetése a GPIO-ra [2.2]. Itt rá kell csatlakoztatni az eszköz vezetékeit, úgy hogy egy eszköz minden SPI vezetéke ugyan azt a csatornát használja.

Amennyiben csatlakoztatva van akkor programon belül meg kell határozni, hogy melyik GPIO van használva melyik célra, itt nem kötelező mind a 4 (MOSI,MISO,CLK,CS) jeleket felsorolni, vagy akár csatlakoztatni fizikailag a MISO esetében. Az ajánlott a MOSI és a CLK jelek, hogy majd lehessen változtatni az SPI módokat.

Beépítetten 8 és 16 bites adat tömböt vár szoftver oldalon, viszont csak az első 8 bitet küldi el, ha a programozó nem állítsa át. Nagyobb adatot is át lehet küldeni (pl. 24bit amit a DAC használ), viszont azt a programozó kell megoldja, mint egy 16 bites és egy 8 bites csomag. Ezért nem használtam a CS jelét, mint SPI jel, hanem digitális jel, mert így teljes kontrollom volt felette és nem kellett féljek, hogy ilyenkor a CS belezavar a küldésbe.

### 4.1.4. Kijelző vezérlése

A kijelző vezérlése a [3.2] ábrán látható osztály diagrammot használja, az SPI osztálynak meg kell adni, hogy hová vannak csatlakoztatva a vezetékek, ezek az adatok megtalálhatóak a **Global.h** fájlban, ezen kívül a kijelző használ még egyedi vezérlő jeleket is, amelyek globálisan definiálva vannak, így ezeket nem kell megadni.

A kijelzőnek van egy DC jele, ami a parancs és adat mód váltásáért felelős. Amennyiben ez a jel magas akkor a beérkező adat parancsként van értelmezve, amíg ez alacsony akkor adatként.

Nincs a teljes képernyő adata elmentve a memóriában egy időben, mivel a kijelzőnek van saját memóriája, így csak módosítani kell azt. A memóriában egy időben csak egy 8 pixel magas sor van eltárolva, mivel egy karakter is ilyen magas. Amint megvan egy sor kiszámítása akkor azt elküldi és kezdi a következő sort. Mivel a kijelző nagy frekvenciás órajellel is képes működni (68Mhz), így a küldési idő is alacsony.

Az ILI9341 osztály csak az inicializálásért felel, a parancs és adat módokban való küldésért és a képernyő teljes újratöltésére egy színnel, vagy a jelenlegi sortól kezdve a kijelző végéig.

A CHARACTERDISPLAY osztály képes egyszerű ASCII alapú üzenetek kijelzésére. Ezt egy mask tömbbel éri el, minden karakternek van egy 8x8as maskja, amely bittenként tárolja el, hogy hol kell változtatni a színt. Ahol a mask 1-es értékű, ott a pixelt a karakter színűre kell festeni, ahol 0 ott a háttérszínűre. Ez minden betűn egyenként végig halad és beteszi a tároló tömb megfelelő indexére a szín értéket, a tároló sor küldés kezdetekor fel van töltve a háttérszín értékével, ez törli az előző sor értékeit és így csak ott kell módosítani ahol szükséges. Küldés során printLine(string) függvény kap egy string értéket. Itt figyelemmel kell lenni arra, hogy automatikusan nem kezd új sort ha hosszú a szöveg. Ezen karakterenként végig haladva átalakítja a karaktereket maskokká és beírja a tárolóba. A

```
void CHARACTERDISPLAY::insertChar(uint8_t position, const uint8_t *charSet)
{
    for (int bit = 0; bit < 8; bit++)
    {
        uint8_t mask = 128 >> bit; // from last in the mask to the first
        for (uint16_t i = 0; i < lineHeight; i++)
        {
            if (charSet[i] & mask)
            {
                // id is the position of the pixel
                int id = i * lineWidth + position * 8 + bit;
                row[id] = fg_Color; // set pixel to fg_Color
            }
        }
    }
}
```

A GRAPHDISPLAY képes használni a CHARACTERDISPLAY függvényeit és kibővíti azzal, hogy lehetővé teszi XY grafikonok kirajzolását. Itt csak egy vektort kell megadni, az szerint lesz skálázva, hogy a kijelzőt a legnagyobb mértékben kihasználja. A kijelző legfelső sorában lesz a maximális érték és az szerint lesz skálázva a többi érték. A kapott értékek számától függően az OX tengelyen az szerint lesznek egyenletesen felosztva az értékek.

A függvény hívásakor meg kell adni, hogy az értékek milyen mértékegységek, ez 2 karakter lehet maximum (pl. mA).

#### 4.1.5. Analóg kapcsoló

Ennek az osztálynak a feladata az analóg kapcsolók vezérlése és ez az osztály vezérli a DAC-ot is. Ez képes egyszerre vezérelni minden hármat kapcsolót egyszerre és mivel a komponens azonosítására nem szükséges analóg jeleket használni, így egy beállítás táblázatot használ, amellyel megadható, hogy melyik ellenállást kapcsolja és, hogy milyen feszültséget adjon le a DAC a következő táblázat szerint [4.1].

Mód	Használt ellenállás	Feszültség
0	Nincs	0V
1	Kicsi(100 Ohm)	0V
2	Kicsi(100 Ohm)	3.3V
3	Közepes(330 Ohm)	0V
4	Közepes(330 Ohm)	3.3V
5	Nagy(4700 Ohm)	0V
6	Nagy(4700 Ohm)	3.3V

4.1. ábra. Kapcsolási táblázat

Így a mérés során csak egy 0-6 közti értéket kell küldeni ahhoz, hogy a megfelelő ellenállást és feszültség legyen minden kapcsoló kimenetén. Lehetőség van arra is, hogy csak a feszültség legyen beállítva és a kapcsoló nyitott állapotban legyen, ez abban az esetben fontos ahol időben való változást kell figyelni, mint például a tranzisztor töltése. Ebben az esetben előre beállítható a DAC kimenete és mivel az ADC osztály a második processzor magon fut így a szemafor jelzésére az ADC elkezd mérni, viszont ha ekkor kerül csak elküldésre az adat a DAC-nak akkor nagy késések lehetnek és az ADC hibás adatokat fog mérni. Így ha csak a kapcsolót kell átkapcsolni akkor a késés minimális és az ADC nem fog hibás adatokat olvasni.

Az a kapcsolók egyszerre való írása maszkolással van elérve. Mivel az analóg kapcsolók egymás mellett helyezkednek el, így azt a részt ki lehet maskolni és az érték maszkba meg azokra a pozíciókra kerül amit be akarunk írni.

```

void ASWITCHCONTROLLER::setSwitchSetting(const uint8_t sw1, const uint8_t sw2, const uint8_t
                                          sw3)
{
    // set switch
    gpio_put_masked(this->mask, 0 | (Sw_translation_Map[sw1].setting << 16) | (
        Sw_translation_Map[sw2].setting << 18) | (Sw_translation_Map[sw3].setting << 20));
}

```

## 4.2. Hardver rész

A rendszer elkészítéséhez szükséges volt egy áramkör tervezése is, mivel több komponens csak SMD (Surface Mount Component) formában találhatóak meg, így nem alkalmazhatóak egy egyszerű breadboard. Ezen megtalálható minden ami szükséges a rendszer működéséhez, csupán egy micro-USB szükséges a rendszer táplálásához.

Az áramkör tervet az EasyEDA-ban terveztem, ez egy ingyenesen használható szerkesztő program, akár webes felületen is használható és nagy mennyiségű alkatrész található meg az adatbázisában amelyek a tervezésre használhatóak.

Az áramkör egy kétoldalas lapon található és csak egy oldalán találhatóak a komponensek. A másik oldalán legfőképpen a huzalozás található. A rendszer használ kis méretű SMD alkatrészeket és THC alkatrészeket is.

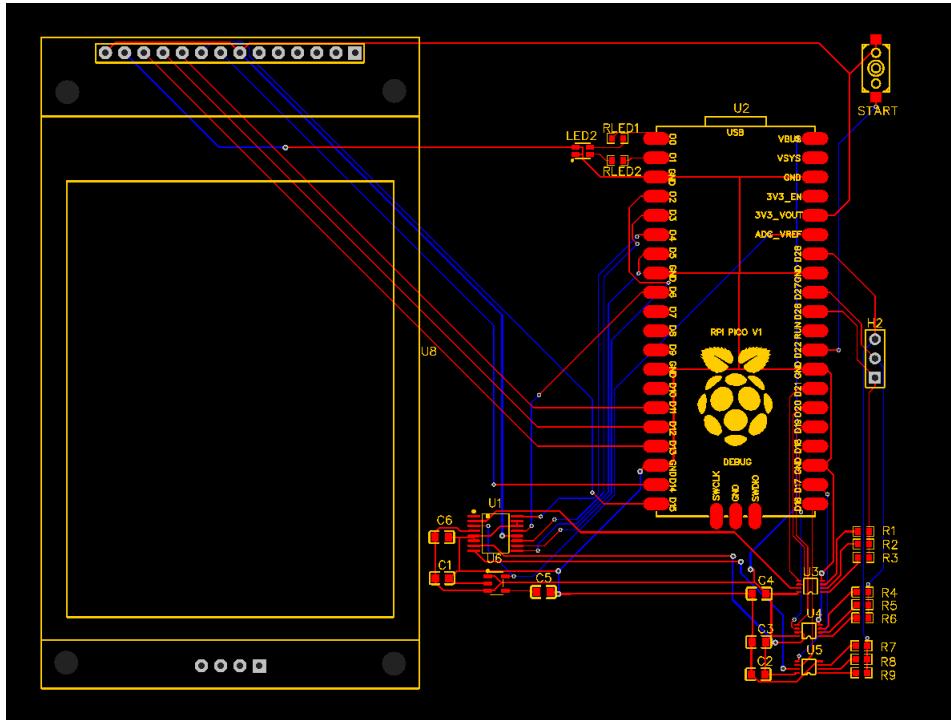
A huzalozás során nagyobb figyelmet fektettem a nagy frekvenciás SPI jelek és a mérőáramkör szétválasztására, a zajok csökkentése érdekében.

A jel vezető huzalok vékonyabbak, mint az áramot vezető huzalok, viszont ez csak design szempontjából van jelentősége, mivel az áramerősség alacsony így vékonyabb huzalok is megfelelnek a feladatra. Egyes esetekben amikor a kis méretű SMD alkatrészekhez kell csatlakoztatni a vezetéket akkor közvetlen a csatlakoztatás előtt a vezeték vastagság lecsökken, hogy lehetséges legyen a forrasztás annálkül, hogy a mellette levő lábba rövidzárt okozzon.

A chipek mellett találhatóak kondenzátorok is, amelyek a tápfeszültség stabilitására szolgálnak.

#### 4.2.1. Problémák az áramkör tervezésekor

Mivel számonomra az áramkör tervezés tárgy nem volt elérhető, így ez volt az első áramkör amit terveztem[4.2]. Így többször át kellett alakítsam, hogy kivitelezhető legyen.



4.2. ábra. NYÁK első verziója

Legelső alkalommal a rendszer nem volt kivitelezhető, mivel túl vékony vezetékeket használtam, ami nem volt legyártható a számonomra elérhető helyeken. A komponenseket nehezen lehetett volna beilleszteni (Through Hole komponensek esetén, mint a kijelző), mivel a vezetékek a lap minden oldalán voltak vezetve, így forrasztás során nehéz lett volna azokat a lábakat forrasztani, amelyek a képernyő oldalán voltak. Ezek át kellett vezessem olyan módon, hogy csak a lap másik oldalán legyen a vezeték csatlakoztatva a komponenshez.

Beszerzés során nem rendeltem meg minden alkatrészt, csak a fontosakat, így néhány egyszerű alkatrészt helyettesítettem más alkatrésekkel, ez csupán egy feszültség referenciát 2 LED-et és egy kapcsolót érintett, így ezeket inkább helyileg helyettesítettem, minthogy ismét rendeljem meg azokat.

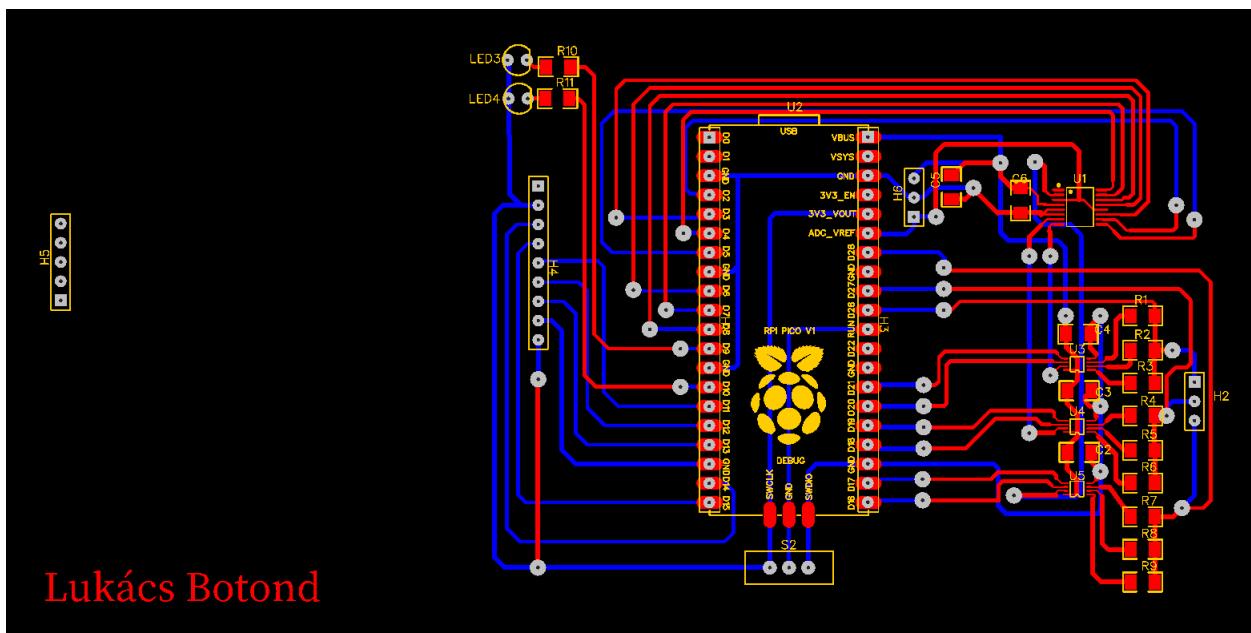
A kijelző mérete a szerkesztőben és a valóságban nem egyezett meg, viszont ez még kiderült az áramkör kinyomtatása előtt, így nem vesztődött el sok idő. Az egész kijelző nagyobb volt, mint a

valóságban, még a láb közei is, így könnyedén látható volt, hogy a tervrajz nem lesz kivitelezhető.

A tervrajzra nem tüntettem fel néhány jelzést, ami a felhasználást segíti, így ezt utólag felrajzoltam a lapra.

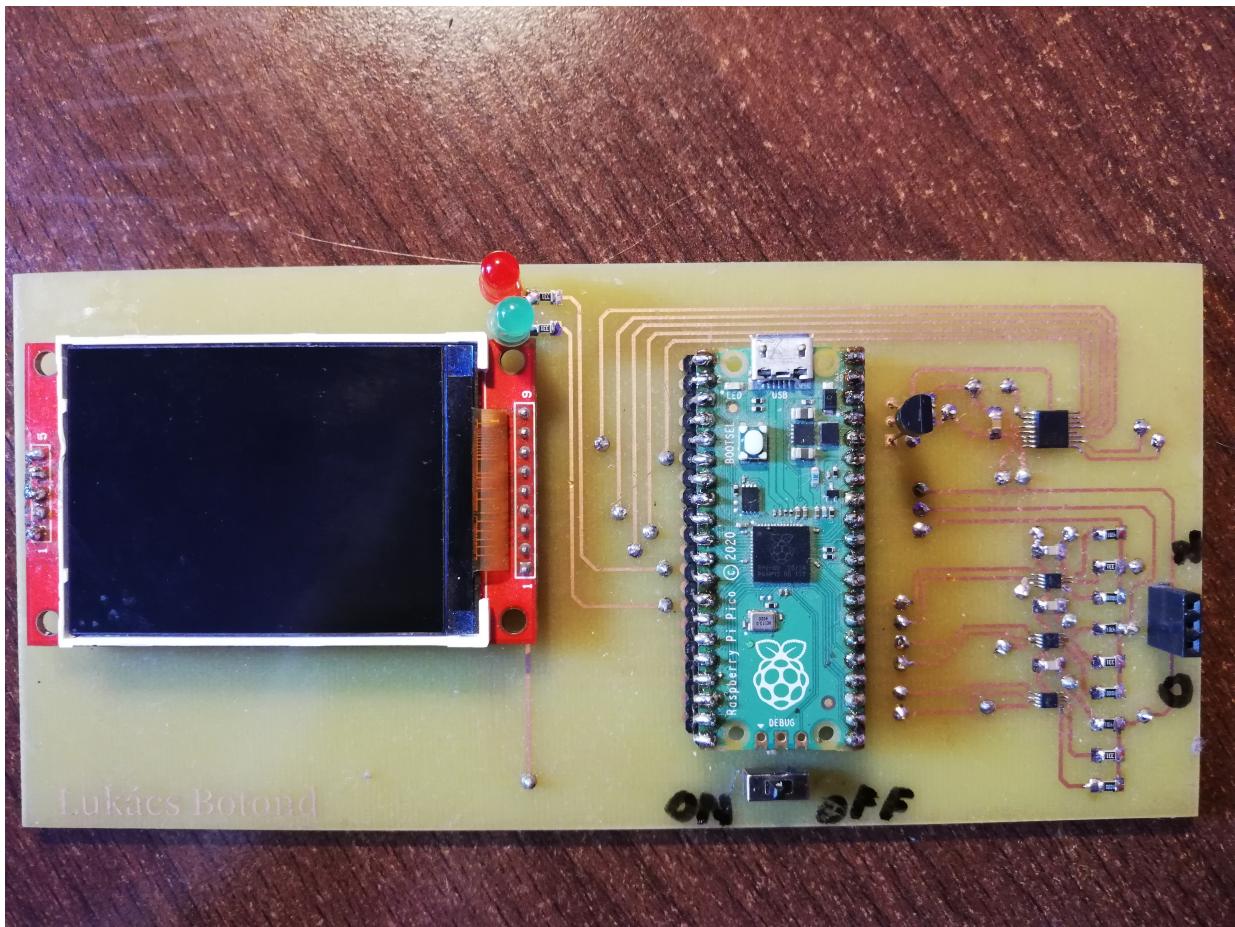
#### 4.2.2. Áramkör elkészítése

Miután a tervezés elkészült és kivitelezhetőnek lett minősítve[4.3] azután az áramkör el lett készítve az egyetemi laboratórium segítségével. Miután a lapot a furatokkal és a réz huzalozással megkaptam azután elkezdtem a komponensek felhelyezését.



4.3. ábra. NYÁK végső verziója

Áramkör forrasztásával szintén nem volt sok tapasztalom, csupán egyszerűbb áramkörökkel, így nem a lehető legszebb, viszont használat során minden elektronikailag csatlakoztatva van, így az áramkör működőképes. A végső áramkör a következőkben néz ki. [4.4]



4.4. ábra. A kész áramkör

### 4.3. Szimulációk

A szimulációra az LTspice [5] alkalmazást használtam, amelyben felépítettem a mérő áramkört és ellenőriztem, hogy az elméleti mérési módszer alkalmas-e a mérés elvégzésére, és az értékek milyen tartományban vannak, mivel a rendszer nem elégé érzékeny, hogy a precíz értékeket pontosan megadja, így olyan módszer kell ahol a tolerancia nagyobb. Egy ilyen példa az áramerősség mérése amit a port ellenállást használja az áramerősség meghatározására, viszont kis ellenállás esetén a feszültség esés az ellenálláson alacsony áram esetén nem mérhető pontosan. Ezért ilyen esetben nagyobb port ellenállást kell alkalmazni, hogy mérhető legyen a feszültség esés.

A program képes az időben változó feszültségeket is mérni, így meg lehet tudni, hogy milyen idő

intervalumban kell nézni és hogy a komponensek időben mennyire befolyásolják a mérést. Ez legfőképpen a kondenzátorok mérésekor volt fontos, hogy megbizonyodjak a mérések valóságosságáról.

A szimulációk többször is segítséget jelentettek a program hibáinak felfedezésében amit annélkül nehezebb lett volna felfedezni.

## **5. fejezet**

### **Üzembe helyezés és kísérleti eredmények**

## **6. fejezet**

### **A rendszer felhasználása**

## **7. fejezet**

### **Következtetések**

# Irodalomjegyzék

- [1] „DAC.” <https://www.ti.com/cn/lit/ds/symlink/dac8565.pdf?ts=1654348067115>. [Online 2022].
- [2] M. Frejek, „AVR Transistortester.” [https://www.mikrocontroller.net/articles/AVR\\_Transistortester](https://www.mikrocontroller.net/articles/AVR_Transistortester). [Online 2022].
- [3] „Arduino Atmega328p datasheet.” [https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P\\_Datasheet.pdf](https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf). [Online 2022].
- [4] U.Tietze-Ch.Schenk, *Analóg és digitális áramkörök 5. javított kiadás (642–645)*. Műszaki könyvkiadó, 1990.
- [5] „LTspice.” <https://www.analog.com/en/design-center/design-tools-and-calculators/Ltspice-simulator.html>. [Online 2022].

## **A. függelék**

### **Függelék**