Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Mesterséges Intelligencia és Rendszertervezés tanszék

Kárpáti Péter Milán

EGYEDI SZOFTVER ÉS HARDVER KÉSZÍTÉSE MODERN KÁVÉFŐZŐ GÉPHEZ

KONZULENS

Scherer Balázs

BUDAPEST, 2024

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Kárpáti Péter Milán**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2024. 12. 08.

...…….
Kárpáti Péter Milán

Összefoglaló

Háztartási gépeink egyre fejlettebb hardver- és szoftvermegoldásokat igényelnek. A kávéfőzők piacán az utóbbi időben olyan modellek jelentek meg, amelyek rendelkeznek hőmérséklet- és nyomásszabályozással. A szakdolgozatom célja egy hagyományos kávéfőző gép átalakítása úgy, hogy képes legyen a hőmérséklet, a nyomás és a lefőzött kávé mennyiségének szabályozására.

A tervezés előtt áttekintettem egy korszerű kávéfőző általános felépítését, főbb komponenseit és azok működési elveit. Ezt követően megvizsgáltam a hőmérséklet, nyomás és súlymérés különböző technológiáit.

A tervezési folyamat kezdetén meghatároztam a rendszerrel szemben támasztott követelményeket, majd ezek alapján kialakítottam a rendszer architektúráját. A tervezett megoldás egy elosztott rendszer, amely egy Raspberry Pi-ra és egy STM32 mikrovezérlőre épül.

Ezt követően a rendszerhez tervezett nyomtatott áramköri lap és egyedi mechanikai alkatrészek tervezése és gyártása következett. Végezetül a szoftver implementálását végeztem el, amely magában foglalta a felhasználói felület elkészítését a Qt keretrendszer segítségével, a szabályozások és mérések megvalósítását a mikrovezérlőn, továbbá a két eszköz közötti kommunikáció megtervezését.

Abstract

Our household appliances require increasingly more advanced hardware and software solutions. Recently, temperature and pressure-controlled coffee machines have appeared on the market. The aim of my thesis is to modify a simple coffee machine to regulate temperature, pressure, and brew volume.

Before starting the design process, I analyzed the general structure of a modern coffee machine, its main components, and their operating principles. Following this, I examined in various methods of temperature, pressure, and weight measurement.

At the beginning of the design process, I defined the system requirements and developed the system architecture based on these requirements. As a result, the solution became a distributed system built on a Raspberry Pi and an STM32 microcontroller.

Next, I proceeded with the design and manufacturing of the printed circuit board and custom mechanical components for the system. Finally, I completed the software implementation, which included creating the user interface on the Raspberry Pi, implementing the control and measurement functions on the STM32 microcontroller, and designing the communication between the two devices.

1 Bevezetés

Manapság a háztartási gépek egyre összetettebb hardveres és szoftveres megoldásokat igényelnek, különösen, ha magas szintű felhasználói élményt szeretnénk elérni. Az elmúlt évek során a baristák körében nagy népszerűségre tettek szert azok a kávéfőző gépek, amelyek képesek a hőmérséklet és a nyomásszabályozására. Projektemhez inspirációt a Decent Espresso kávéfőző nyújtott, amelyet az elsők között láttak el ilyen funkciókkal. Az ilyen gépek ára azonban jellemzően igen magas. Szakmai érdeklődésemből adódóan ez arra ösztönzött, hogy elgondolkodjak, vajon át tudnám-e alakítani a saját otthoni kávéfőzőmet úgy, hogy hasonló funkciókkal rendelkezzen.

A kávé extrakciója az a folyamat, amely során a forró víz segítségével kivonjuk a kávéőrleményből az íz és aromaanyagokat. Ez a folyamat alapvető fontosságú a kávékészítés során, mivel meghatározza a ital végső ízét és minőségét. A megfelelő extrakció biztosítja, hogy a kávé kiegyensúlyozott legyen, elkerülve a túlzott keserűséget vagy savanyúságot. Az extrakciós folyamatot számos tényező befolyásolja:

- őrlés finomsága (espresso esetén 200-400 mikron)
- vízhőmérséklet (espresso esetén ~90-96 °C)
- kávé víz aránya (espresso esetén ~1:2)
- főzési idő (espresso esetén 25-35 s)

Ezeknek a tényezőknek a megértése és szabályozása segíthet abban, hogy kihozzuk a maximumot egy adott kávéból és ízlésünkhöz igazítsuk. A kávébabokat a szüret és feldolgozás után különböző mértékben pörkölik, jellemzően a világos pörkölésű kávék magasabb, míg a sötétebb pörkölésű kávék alacsonyabb hőmérsékletet igényelnek. A professzionális eszpresszógépek túlnyomórészt 9 bar körüli nyomáson működnek, ugyanakkor az utóbbi időben szűk körben nyomás profilok alkalmazásával is kísérleteznek. Fontos, hogy a 9 bar nyomást ne lépjük túl, mert magasabb nyomások esetén a kávé korongban apró csatornák keletkeznek, melyek egyenetlen extrakcióhoz vezetnek.

A projekt fő célkitűzése az volt, hogy a kávéfőzési hőmérséklet 1 °C, a nyomás 0,5 bar pontossággal szabályozható legyen. Továbbá, a fő funkciók és paraméterek egy érintő kijelzőről állíthatóak legyenek. Ezen kívül a tervek között szerepelt egy mérleg

beépítése a csepegtető tálcába, melynek segítségével a lefőzni kívánt kávé mennyisége is állítható.

A feladatot egy általam tervezett egyéni hardverrel valósítottam meg, amelyen egy Rapsberry Pi Zero 2W és egy STM32F411-es mikrovezérlő található. A rendszerben a felhasználói felület és fő állapotok kezelését az RPi modul, míg a szabályozási és alacsony szintű feladatokat az STM32-es mikrokontroller végzi. A mérleg mechanikus kialakításához 3D modelleket készítettem, amelyek 3D nyomtatással kerültek elkészítésre.

A dolgozat második fejezetében áttekintem egy modern kávéfőző felépítését. Bemutatásra kerül a szivattyú és bojler működése, a víz áramlási útja, illetve a beépített biztonsági mechanizmusokra is kitérek.

A harmadik fejezetben a folyamat jellemzőinek mérési módszereit ismertetem. Ennek keretében bemutatom a hőmérsékletmérés különböző módszereit, a legyakrabban előforduló nyomásszenzorok, erőmérő cellák működését.

A negyedik fejezetben a projekt során felhasznált technológiákat mutatom be, részletezésre kerül az I^2C kommunikációs protokoll, valamint a FreeRTOS valós idejű operációs rendszer.

Az ötödik fejezet a rendszer központi moduljainak kiválasztását, a választási szempontokat és a végső döntés okait tárgyalja. Itt egy blokkvázlat segítségével bemutatom a megvalósított rendszer felépítését, valamint az egyes elemek szerepét.

A hatodik fejezetben a nyomtatott áramkör és a 3D modellek tervezési folyamatát ismertetem. Az első részben a kapcsolási rajz, a másodikban pedig a huzalozási terv készítésének folyamatáról írok. A fejezet zárásaként bemutatom a beépített mérleg mechanikai megvalósítását.

A hetedik fejezetben a mikrokontrolleren futó program fejlesztési eszközeit és főbb szoftverkomponenseit ismertetem. A fejezet végén a Qt keretrendszert, valamint a Raspberry Pi-on futó szoftvert mutatom be.

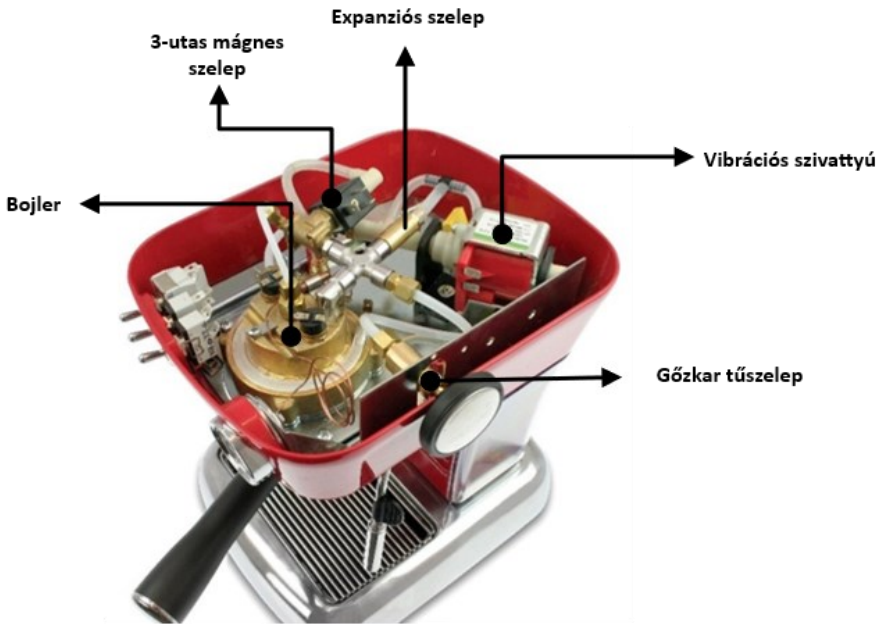
A nyolcadik fejezteben a megvalósított szabályozások és mérő eszközök eredményeinek kiértékelését prezentálom.

A kilencedik fejezetben összefoglalom az elvégzett munkát, valamint felvázolom a jövőbeli fejlesztési terveket.

2 Kávéfőző felépítése

Ebben a fejezetben a kávéfőző felépítését, főbb alkatrészeinek működését és a különböző működési módokat fogom bemutatni.

Az átalakításra kiválasztott kávéfőző egy Ascaso Dream 2010-es modell, amely ideális alapot nyújt a projekthez. Kiváló minőségű anyagokból és alkatrészekből épül fel, de nem tartalmaz nyomtatott áramköri lapot és digitális alkatrészeket. Kihívást jelent azonban a rendelkezésre álló hely, ami jelentős megkötést jelentetett tervezés során.

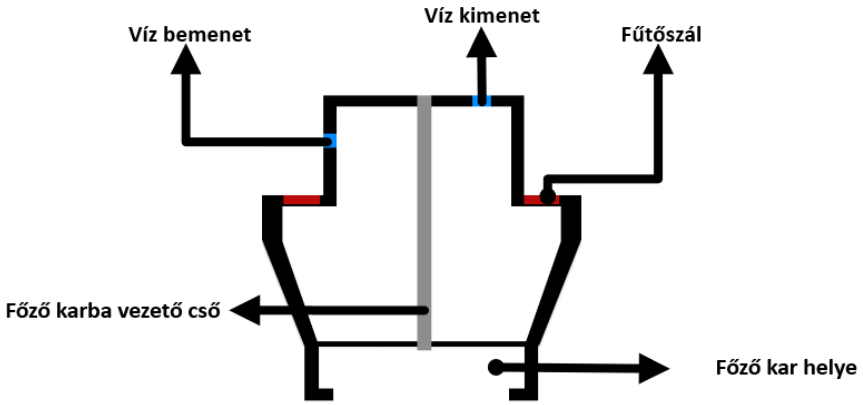


2.1. ábra Ascaso Dream kávéfőző [1]

A gép minden szükséges biztonsági funkcióval el van látva, amelyek akkor is megbízható védelmet nyújtanak, ha az általam telepített szabályozó berendezés meghibásodik. A bojler két bimetall hőkapcsolóval rendelkezik, melyek az átalakítás előtt a kétállású hőszabályozásért feleltek: az egyik a főzési, míg a másik a gőzölési hőmérsékletet szabályozta. Ezen kívül van egy vészhelyzeti hőbiztosíték, ami kiold, ha a hőmérséklet veszélyes szintet ér el, de ekkor a biztosíték cserére szorul. Továbbá az eszköz rendelkezik egy expanziós szeleppel, amely túlnyomás ellen nyújt védelmet.

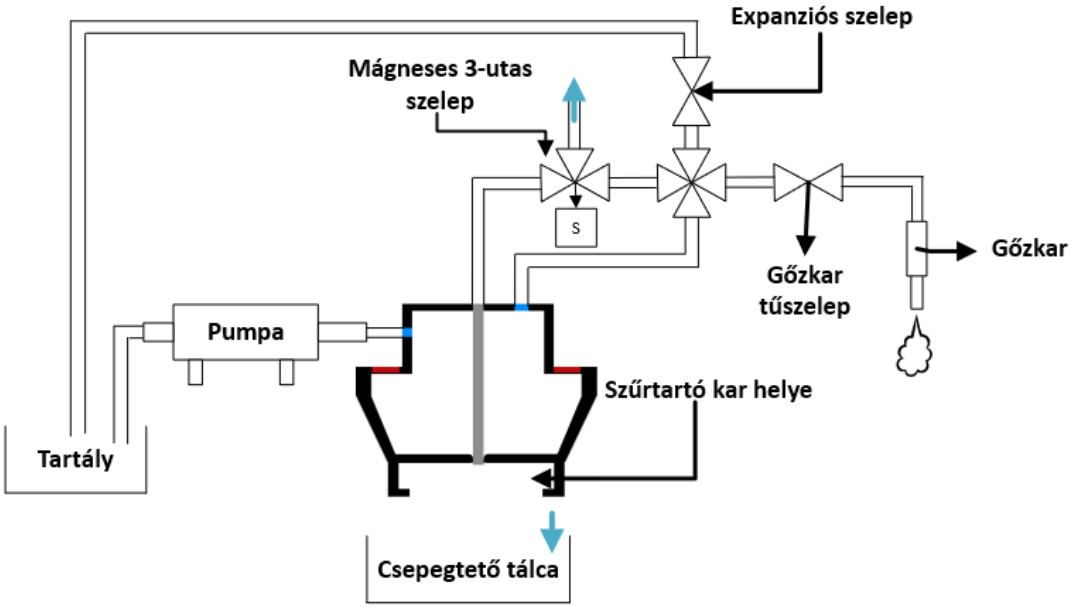
2.1 Vízkör

A gép központi eleme egy rézbojler, melynek felépítése rendkívül egyszerű. A víz melegítését egy, a bojler házába mélyített fűtőszál végzi. Emellett rendelkezik egy bemeneti és egy kimeneti nyílással és egy átvezető csővel, amelyen keresztül a nagynyomású víz a szűrőkarba jut.



2.2. ábra Ascaso Dream bojler sematikus ábrája

Az alábbi ábrán a kávéfőző vízrendszerének kapcsolási rajza látható, amely szemlélteti a gép főbb elemeinek összeköttetéseit és a víz áramlási útjait a különböző működés állapotokban.



2.3. ábra A kávéfőző hidraulika sematikus ábra

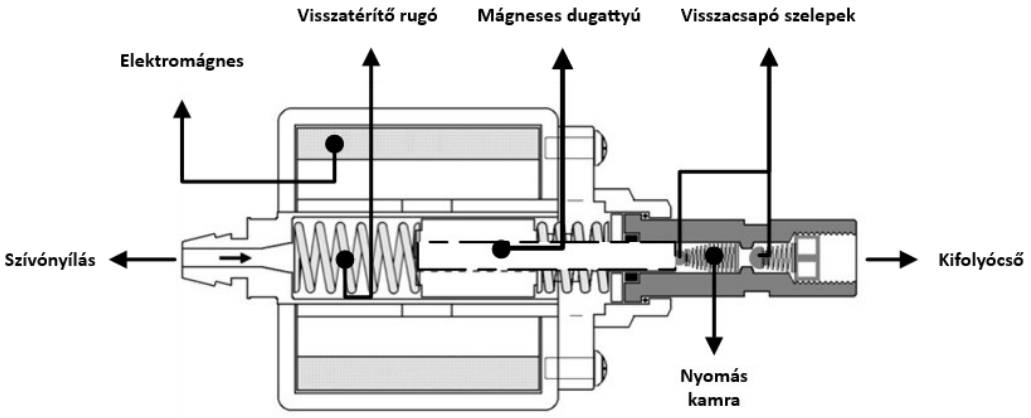
A szivattyú beindításával a bojler megtelítődik vízzel és a tetején található kékkel jelölt kimeneti nyilason keresztül egy 4-es elosztó csatlakozásba jut. Innen a szelepek állapotától függően több irányba áramolhat a víz:

1. Kávéfőzéskor: A 3-utas mágnesszelepet és a gőzkar tűszelepét zárva tartjuk, ekkor a víz a 3-utas szelep egyenes ágán keresztül a bojlerbe, majd a szűrőkarba jut.
2. Kávéfőzés után: Főzés után a szűrőkarban túlnyomás uralkodik, ekkor a 3-utas mágneses szelepet kinyitva, a víz a csepegtető tálcába folyik.
3. Gőzöléskor: A tűszelepet kinyitjuk, és a 3-utas szelepet zárva tartjuk. Ebben az esetben a bojlert magasabb hőmérsékletre szabályozzuk, így vízgőz keletkezik.
4. Biztonsági nyomásleeresztés: Ha minden szelep zárva van esetleg valamelyik eltömődik és a nyomás túllépi a 15 bart, az expanziós szelep kinyílik.

Számos egyéb kombináció lehetséges, de ezek a valós használat során nem relevánsak, így ezeket nem részletezem.

2.2 Vibrációs szivattyú

A legtöbb modern kávéfőzőben vibrációs szivattyút használnak. Ennek oka, hogy a forgólapátos szivattyúk árának töredékéért beszerezhetőek és kellően magas nyomás előállítására képesek. Hátrányuk, hogy a nyomást nem folytonosan, impulzusosokkal állítják elő, amely a mérést zajossá teszi. Ahhoz, hogy képesek legyünk a nyomást szabályozni fontos megérteni működésüket.



2.4. ábra Rezgő szivattyú felépítése [1]

Az ilyen szivattyúk működése elektromágneses elven alapul. A mozgás két ciklusból áll:

1. Első lépésben, amikor a váltakozó feszültség pozitív, az elektromágnes aktiválódik, és visszahúzza a dugattyút, lehetővé téve, hogy a víz beáramoljon a nyomás kamarába
2. A szivattyú belső diódával van ellátva, amikor a feszültség negatív a tekercsben nem folyik áram és a visszatérítő rugó a dugattyút a kifolyócső felé tolja. A víz visszafolyását a visszacsapó szelepek gátolják, így a víz nagy nyomással távozik a kifolyócsövön.

Ez azt jelenti, hogy a dugattyú a váltakozó feszültség frekvenciájával oszcillál és magas nyomású víz „impulzusokat” hoz létre. Ennek a nyomásszabályozásnál fontos szerepe van, melyről a 7.1.6 fejezetben írok.

3 Mérési módszerek

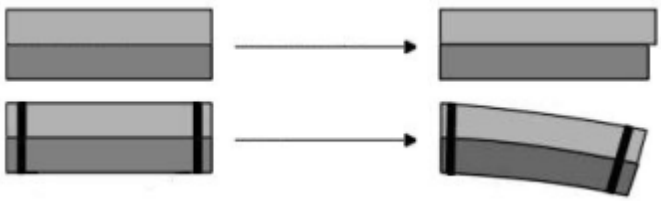
3.1 Hőmérséklet mérés

A hőmérséklet pontos mérése életünk számos területén rendkívül fontos szerepet játszik, mivel a fizikai jellemzők és folyamatok jelentős része függ a hőmérséklettől. Mérésére számos különböző eleven működő módszer létezik, ezek két fő csoportra bonthatóak: a fizikai testtel közvetlen érintkezésbe kerülő és az érintés mentes hőmérők.

Minden, abszolút nulla hőmérsékletnél melegebb test elektromágneses sugárzást bocsájt ki az infravörös tartományban. Az érintésmentes hőmérők ezt, a test által kibocsájtott sugárzást mérik. Azonban ezek méretük és komplexitásuk miatt nem praktikusak egy háztartási gépbe, ráadásul pontosságuk függ az anyag emissziós tényezőjétől. Ezért a továbbiakban az érintkezéses hőmérők leggyakrabban használt módszereire fókuszálok.

3.1.1 Bimetall

A hőmérsékletmérés két egymáshoz rögzített, eltérő hőtágulási tényezőjű fém hőmérsékletváltozás okozta deformációján alapul [3]. Ha az egyik vég rögzített, akkor a hőmérsékletváltozás hatására a bimetall meghajlik. Általában előre beállított hőmérséklet elérésekor aktuátorként használják kapcsoló működtetésére.

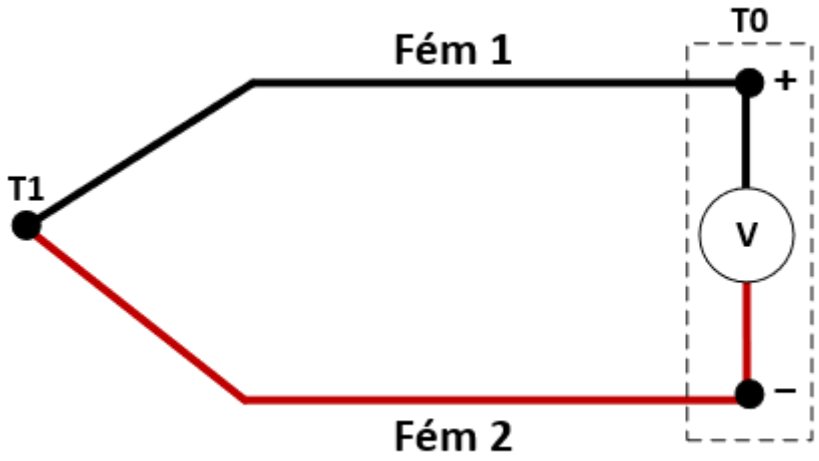


3.1. ábra Bimetall szemléltető ábra [3]

Előnye, hogy rendkívül költséghatékony, és nem igényel karbantartást, így nagyon megbízható megoldást nyújt. A legtöbb piacon kapható kávéfőző gépben ilyen elemek segítségével valósítanak meg kétállású szabályozást. Ez pontos hőmérséklet-szabályozásra nem alkalmas, de biztonsági elemnek jól használható.

3.1.2 Hőelem

A hőelemek működése a Seebeck-effektuson alapszik. Ha két különböző fém csatlakozási helye (melegpont) és szabad végéi (hidegpont) között hőmérsékletkülönbség van akkor az érintkezési helyen elektromotoros erő lép fel. Így a szabad végek között elektromos feszültég mérhető [3].

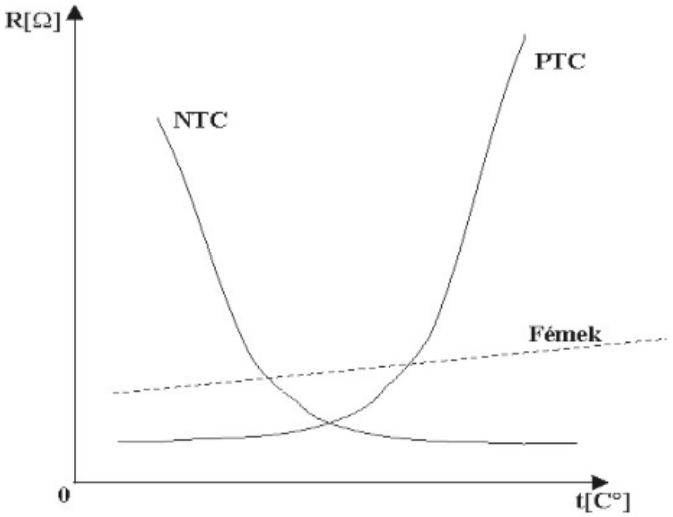


3.2. ábra Seebeck effektus

A hőelemek esetében a hidegponthoz viszonyítva tudjuk meghatározni a melegpont hőmérsékletét. Ez azt jelenti, hogy vagy mérnünk kell a hidegpont hőmérsékletét és hidegpont kompenzációt kell alkalmaznunk vagy ismert hőmérsékletre kell beállítanunk. A hőelemek előnye, hogy széles hőmérsékelt tartományban használhatóak, alacsony áron elérhetőek és mivel hőkapacitásuk általában kicsi gyorsan reagálnak a hőmérsékletváltozásra. Hátrányuk, hogy nemlineáris karakterisztikával rendelkeznek, melynek oka, hogy a Seebeck együttható hőmérséklet függő. További nehézséget jelent, hogy a keletkező feszültség mikrovoltos nagyságrendű így erősítésre és zajszűrésre van szükség. Az ezzel a módszerrel elérhető pontosság a használt anyagoktól és körülményektől függően körülbelül ±0,5 – 2,5 °C.

3.1.3 Termisztor

A termisztorok működése a termoreziztivitás jelenségén, azaz az elektromos ellenállás hőmérséklet-függőségén alapszik. Ezek olyan kerámia félvezetőből készült áramköri elemek, melyek ellenállása a hőmérséklet függvényében számottevően, 0 - 100 ˚C-os tartományban közel két nagyságrendet változik. Két fajtáját különböztetjük meg, az NTC (Negative Thermal Coefficient) típusúakat, melynek a hőmérséklet növekedésével az ellenállása csökken és a PTC (Positive Thermal Coefficient) típusúakat, ezeknek az ellenállása a hőmérséklet emelkedésével nő [4]. A termisztorok erősen nemlineáris karakterisztikájúak, a jellegzetes karakterisztikák az alábbi ábrán láthatóak.



3.3. ábra Termisztorok jelleggörbéje [4]

A módszer előnye, hogy a termisztorok kis méretűek, olcsók és nagy hőérzékenységűek, gyorsan reagálnak a hőmérsékletváltozásra. Hátrányuk, hogy csak nagy szórással gyárthatóak, erősen nem lineárisok, nem energiahatékonyak és szűk hőmérséklet tartományban használhatóak.

3.1.4 Fém hőellenállások

A fém ellenállás hőmérők esetében szintén a termoreziztivitás jelenségét használjuk ki, azonban a termisztorhoz képest az ellenállás hőmérséklet függése jelentősen kisebb. A fémek pozitív hőmérsékleti együtthatóval rendelkeznek, azaz ellenállásuk a hőmérséklet emelkedésével nő. Pontos mérésekhez leggyakrabban platinából készült szenzorokat alkalmaznak. Ennek legfőbb oka, hogy rendkívül jó a linearitása, széles a hőmérséklet-tartománya és mivel nemesfém, ezért nem korrodál [3].

Az iparban leggyakrabban a PT100-as ellenállásokat használják, melyeknek ellenállása 0 °C-on 100 Ω. A DIN EN-60751 szabvány szerint ilyen szenzorok ellenállása az alábbi összefüggéssel számítható T > 0°C esetén:

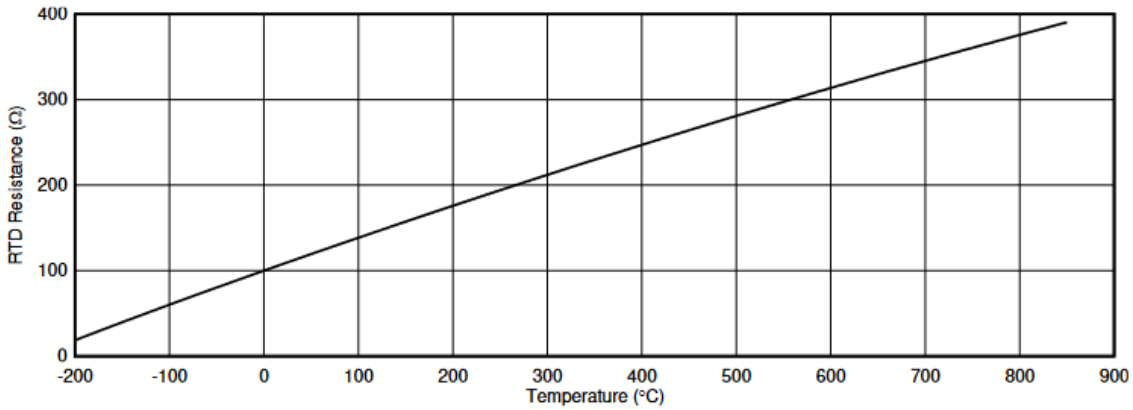
$$T < 0\,°C\colon\ R(T) = R(0) \cdot [1 + a \cdot T + b \cdot T^2 + c \cdot T^3 \cdot (T - 100)]$$

$$T > 0\,°C\colon\ R(T) = R(0) \cdot [1 + a \cdot T + b \cdot T^2]$$

$$a = 3.9083 \cdot 10^{-3}$$

$$b = -5.775 \cdot 10^{-7}$$

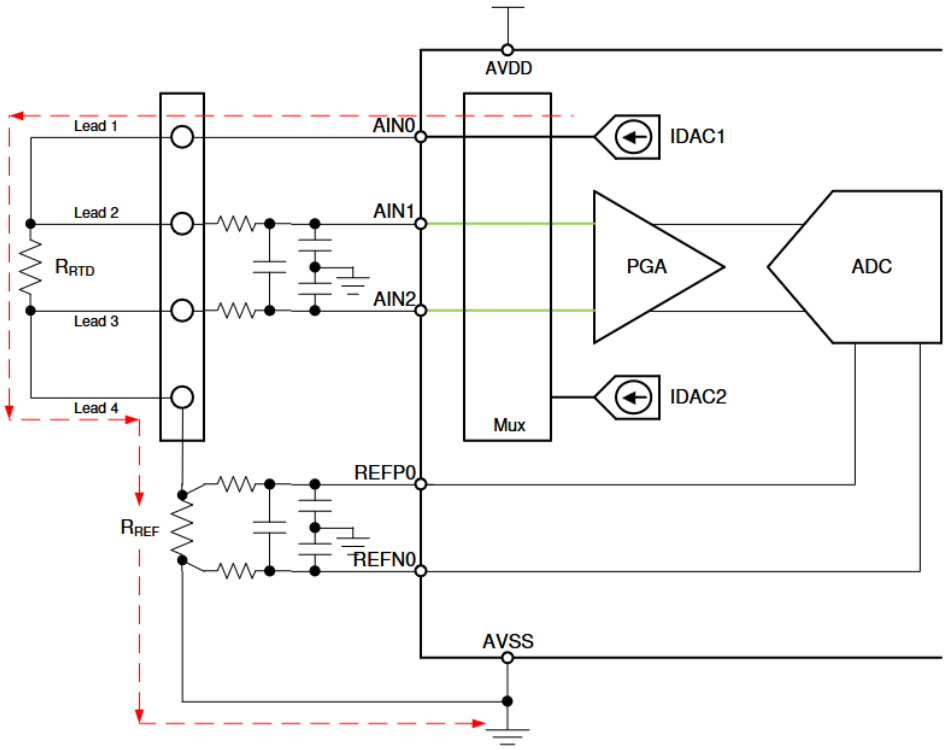
$$c = -4.183 \cdot 10^{-12}$$



3.4. ábra PT100 karakterisztikája [5]

Az összefüggésből látható, hogy a nemlineáris komponensek együtthatói nagyságrendekkel kisebbek, mint a lineáris együttható. Ez azt jelenti, hogy lineáris közelítést alkalmazva, egy PT-100-as ellenállása 1°C-os hőmérséklet változás esetén 0.3908 Ω-al nő. Számunkra a hasznos információt az ellenállás megváltozása hordozza, amely kicsi az alap ellenálláshoz képest. Ez csökkentheti az AD átalakítás után a hasznos felbontást. További nehézséget jelent a mérővezeték ellenállása, amely szintén hőmérsékletfüggő, így az általa okozott hibát nem lehet egy statikus eltolással kompenzálni.

Ezeket a hibákat a megfelelő mérési összeállítás segítségével kiküszöbölhetjük. A feladatra 4-vezetékes mérési módszert választottam. Érdemes megjegyezni, hogy ezt az összeállítást akkor szokták alkalmazni, ha nagyon pontos mérésekre van szükség, az én felhasználásomhoz ennél egy egyszerűbb is megfelelő lenne.



3.5. ábra 4-vezetéskes hőmérséklet mérés [5]

A mérési konfigurációban egy konstans áramforrást alkalmazunk, általában 1 mA árammal. Fontos, hogy kis áramot használjunk, hogy az önmelegedésből adódó hibát minél kisebbre redukáljuk. A PT100-as érzékelőt egy precíziós ellenállással kötjük sorba, amely az A/D átalakító referenciájaként szolgál, így az áramforrás pontatlanságából eredő hibákat kiküszöbölhetjük. A hőellenálláson a feszültséget két külön vezetéken mérjük differenciálisan. Mivel a mérővezetékeken nem folyik áram, így nem esik rajtuk feszültség sem, ezáltal megszűnik a mérővezetékek ellenállásából származó hiba.

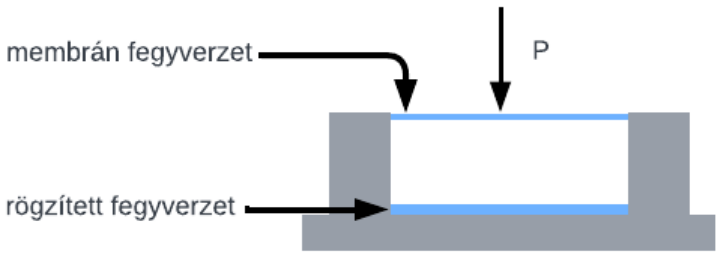
3.2 Nyomás mérés

A nyomásmérő szenzorok két fő csoportra bonthatóak: mechanikus és elektromos elv alapján működőekre. Ebben a dolgozatban kizárólag az elektromos elven működő nyomásmérő szenzorokat tárgyalom.

A nyomás definíció szerint egy adott felületre ható erő nagyságának és a felület nagyságának hányadosa. Mérésére számos módszer létezik a dolgozat keretében a kapacitív és piezorezisztív, módszerek működéséről írok.

3.2.1 Kapacitív

A kapacitív nyomásmérő szenzorok működése az elektromos kapacitás megváltozásán alapul. Az ilyen szenzorokban két elektromos vezető felület: egy rugalmas, deformálódó membrán és egy rögzített lemez, egy kondenzátort alkotnak.



3.6. ábra Kapacitív nyomásmérő szenzor [6]

Ha a rendszert síkkondenzátorral közelítjük, akkor a kapacitás az alábbi egyenlettel írható le:

$$C = \varepsilon_0 \varepsilon_0 \cdot \frac{A}{x}$$

ahol $\boldsymbol{\varepsilon_0}$ az abszolút dielektromos állandó, $\boldsymbol{\varepsilon_r}$ a dielektrikum relatív dielektromos állandója, **x** a fegyverzetek közötti távolság, $\boldsymbol{A}$ pedig a fegyverzetek felülete.

Az egyenletből jól látszik, hogy a kapacitás fordítottan arányos a fegyverzetek távolságával. A nyomáskülönbség hatására a membrán deformálódik, így a relatív kapacitás változás a közelítést alkalmazva egyenesen arányos a nyomás változással. Ugyanakkor valós alkalmazásokban a nem egyenletes deformációból és a szórt kapacitásokból adódóan az ilyen szenzorok nem rendelkeznek jó linearitással [3] [7].

A módszer előnye, hogy kis elmozdulások mérésére is alkalmas, mechanikusan egyszerű és megbízható. Emellett kis mérteben megvalósíthatóak így alkalmazhatóak például mikromechanikai nyomás mérésre [8].

3.2.2 Piezorezisztív mérés

A mérés alapja, hogy mechanikai feszültség hatására a vezető és félvezető anyagok ellenállása megváltozik. A változás a tenzometrikus és piezorezisztív effektus együttes fellépéséből adódik [8]. Az ellenállás a következő összefüggés segítségével írható le:

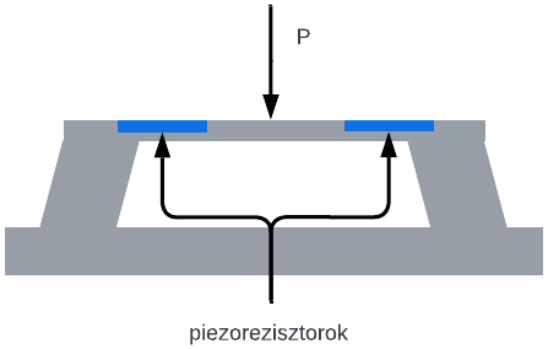
$$R = \rho \frac{l}{A}$$

ahol $\boldsymbol{\rho}$ a vezető fajlagos ellenállása, $\boldsymbol{l}$ a vezető hossza és $\boldsymbol{A}$ a vezető keresztmetszete.

Mechanikai feszültség hatására a vezető alakváltozást szenved: a rúd megnyúlásával egyidejűleg keresztmetszete csökken, ez a tenziometrikus hatás. Ezzel párhuzamosan a mechanikai feszültség hatására a fajlagos ellenállás is változik, ami a piezorezisztív hatás következménye. A két hatásból adódik a bélyegállandó, amely azt mutatja meg az egységnyi fajlagos nyúlás hatására a vezető mekkora fajlagos ellenállásváltozást szenved. [8].

$$g = \frac{\frac{\Delta R}{R}}{\frac{\Delta l}{l}}$$

A piezorezisztív méréshez általában félvezető anyagokat használnak, ebben az esetben a tenziometrikus hatás közel elhanyagolható.



3.7. ábra Piezorezisztív nyomás mérés [6]

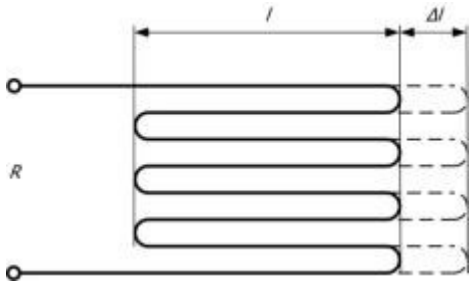
A piezorezisztív szenzorokban a nyomás hatására fellépő ellenállásváltozások kicsik, ezért a pontos mérések érdekében Wheatstone-hidat alkalmaznak. A projekt megvalósításához egy XDB401-es piezorzisztív szenzort választottam, kedvező ára, megfelelő mérési tartománya és pontossága miatt [9].

3.3 Súlymérés

Leggyakrabban mérlegekhez erőmérő cellákat más néven mérlegcellákat alkalmaznak, működésük alapját a nyúlásmérő bélyegek képezik.

3.3.1 Nyúlásmérő bélyeg

A nyúlásmérő bélyeg egy vékony szigetelőlapra “cikk-cakk” mintázatban felvitt ellenálláshuzalból áll. A mérés elve ugyan azonos a piezorezisztív nyomásmérő szenzorokéval, de míg a piezorezisztív érzékelők általában félvezetőből készülnek, addig nyúlásmérő bélyegek esetén fémes vezetőket használnak. Ezért ebben az esetben a tenziometrikus hatás, azaz a deformáció következtében létrejövő ellenállásváltozás lesz domináns. A szigetelőlapot a mérendő felülethez ragasztják, így a bélyeg megnyúlása megegyezik a mérendő felület megnyúlásával. A pontos mérés érdekében a ragasztásnak rendkívül erősnek kell lennie, míg a szigetelőlapnak elég rugalmasnak ahhoz, hogy ne befolyásolja a mérendő test alakváltozását.

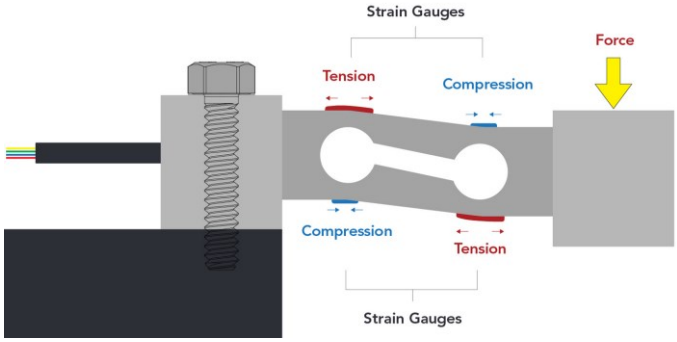


3.8. ábra Nyúlásmérő bélyeg huzalozása [8]

Az ellenállásváltozás ebben az esetben is kis mértékű, ezért a méréshez Wheatstone-hidas összeállítást alkalmaznak.

3.3.2 Erőmérő cella

Az erőmérő cellák a mechanikai erőt elektromos jellé alakítják, leggyakrabban nyúlásmérő bélyegek segítségével. A cellák precízen megmunkált fémtestek, amelyeket úgy terveznek meg, hogy terhelés hatására meghatározott pontokon kiszámítható módon hajoljanak meg. A hajlási pontokon elhelyezett nyúlásmérő bélyegek az erőhatásra deformálódnak, miközben ellenállásuk megváltozik. A cellákon legtöbbször közvetlenül elhelyezik Wheatstone-hidat, jó minőségű cellák esetén a nullpontot és hőmérséklet kompenzálást további ellenállásokkal állítják be [3].



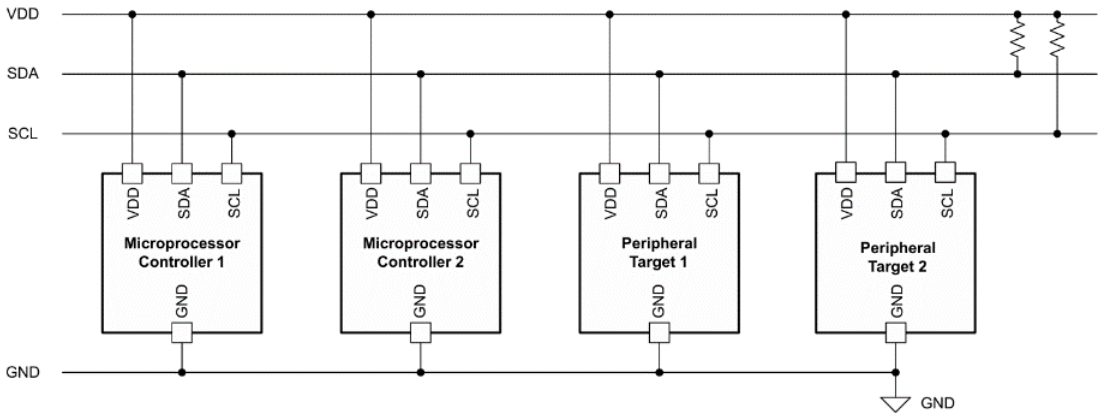
3.9. ábra Erőmérő cella működése [10]

Mérlegek esetén általában egy-pontos erőmérő cellákat alkalmaznak. Ennek oka, hogy az ilyen cellák fizikai kialakítása lehetővé teszi, hogy excentrikus terhelések esetén is kis hibával mérjenek. Ezzel szemben más típusú cellák, a terhelés pontos elhelyezését igénylik a mérési eredmények pontosságának biztosítása érdekében.

4 Felhasznált technológiák

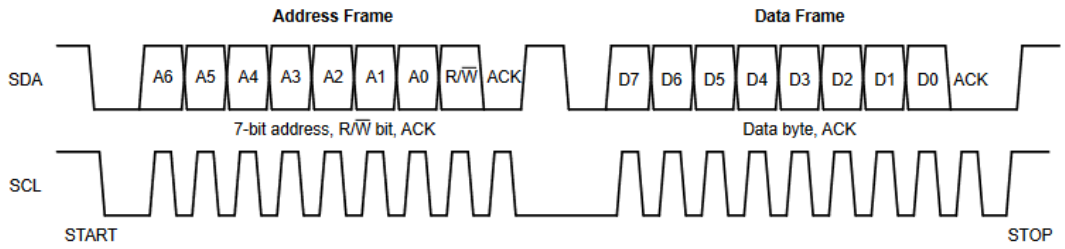
4.1 I²C

Az I²C vezetékes egy széles körben használt soros kommunikációs protokoll, amelyet az Philips Semiconductors fejlesztett ki. Legfőbb előnye, hogy a kommunikációhoz csak két vezetékre van szükség és több céleszközt is támogat, illetve egyszerűsége miatt költséghatékony megoldás.



4.1. ábra Tipikus I²C implementáció [11]

A szabványban minden eszköz ugyanahhoz az órajel (SCL) és adatvezetékez (SDA) csatlakozik, nyitott kollektoros kimenetekkel. A nyitott kollektoros kimenet általában egy N-MOSFET-et használ, melynek aktiválásakor a jelet földpotenciálra húzza le. Ha zárt állapotban van akkor a kimenet lebeg ezért a vezetékeket felhúzó elleállásokkal látják el. Több master és több slave is használható, az eszközök számát a busz kapacitása korlátozza. Minden eszköz rendelkezik egy 7 bites címmel, ez lehetővé teszi, hogy adatokat küldjünk specifikusan egy eszköznek [11]. Mivel csak egy adatvezetékünk van ezért a protokoll csak félduplex kommunikációra alkalmas. Az szabvány különböző sebességmódokat támogat, amelyek meghatározzák az adatátvitel sebességét. Az alapértelmezett sebesség a standard mód 100 kbit/s sebességgel. A gyors mód (Fast Mode) 400 kbit/s, míg a High-Speed Mode (HSM) akár 3,4 Mbit/s sebességre is képes, illetve a leggyorsabb Ultra Fast mód 5 Mbit/s-ra. Mint látható, ezek a sebességek viszonylag alacsonyak a modern kommunikációs protokollokhoz képest, ezért tipikusan A/D, D/A átalakítókkal EEPROM-okkal és kis adatátvitelt igénylő szenzorokkal használják.



4.2. ábra I²C kommunikáció menete [11]

A kommunikációt mindig a Master kezdeményezi és szolgáltatja az órajelet. Az indításához nullára húzza először az SDA vezetéket majd az SCL vezetéket, ez a START kondíció. Ezt követően elküldi a 7 bites címet, majd egy további bitet, amely jelzi, hogy írási vagy olvasási műveletet szeretne végrehajtani (0: írás, 1: olvasás). Minden byte után a fogadó eszköz az SDA vezetéket alacsonyra húzza, jelezve ezzel, hogy sikeresen fogadta az adatot (ACK, azaz acknowledge). Ezután tetszőleges számú byte írható vagy olvasható. A kommunikáció lezárásához a Master magasra engedi az SCL vezetéket, majd az SDA vezetéket, ez a STOP kondíció.

4.2 FreeRTOS

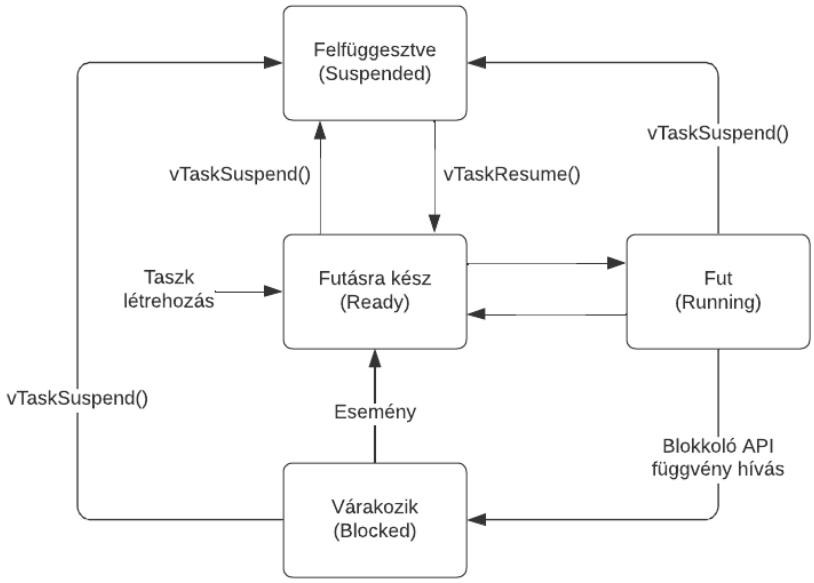
A párhuzamos feladatok egyszerűbb kezelése, érdekében úgy döntöttem, a feladatot egy valós idejű operációs rendszer segítségével oldom meg. Az RTOS (Real Time Operating System) olyan operációs rendszer, melyek célja a kiszámítható, determinisztikus működés és a pontos időzítések elérése.

A projekthez a FreeRTOS-t választottam, ami egy ingyenes és nyílt forráskódú valós idejű operációs rendszer. Előnye, hogy alacsony erőforrás igényű így jól alkalmazható limitált memóriával és teljesítménnyel rendelkező mikrovezérlők esetében. Ebben a fejezetben az ehhez kapcsolódó alapfogalmakat mutatom be:

4.2.1 Taszkok

A teljes feladatot párhuzamosan futó részfeladatokra bontjuk, amelyek önálló egységeknek tekinthetőek, saját memóriaterülettel rendelkeznek. Beágyazott rendszerek esetében jellemzően egymagos processzorokat alkalmazunk, így a processzor egyszerre egy taszkot képes futtatni. Az ütemező szerepe, hogy meghatározza, a processzornak melyik taszkot kell végrehajtania az adott pillanatban. A taszkok az alábbi lehetséges állapotokban lehetnek [12]:

- Fut: A processzor éppen ezt a taszkot hajtja végre. Egy magos processzor esetén egyszerre maximum egy taszk lehet ilyen állapotban
- Futásra kész: Olyan taszk, amely futásra kész, de jelenleg nincs végrehajtás alatt, mert egy vele azonos vagy nagyobb prioritású taszk fut éppen.
- Várakozik: Egy folyamat ebben az állapotban van, ha valamilyen eseményre várakozik. Ez lehet például külső esemény, késleltetés, szemafor. Az ilyen eseményeknek általában van egy időkorlátja amíg vár az eseményre, utána automatikusan futásra kész állapotba lép.
- Felfüggesztve: A várakozáshoz hasonló, azonban ebben az esetben nincs időtúllépés. Ebből az állapotból csak akkor lép ki, ha kifejezetten ezt az utasítást adjuk.



4.3. ábra Taszkok állapotai

4.2.2 Ütemező

Az ütemező azért felelős, hogy eldöntse melyik taszk fusson a következő ciklusban. Ennek megvalósítására különböző ütemezési algoritmusok állnak rendelkezésre. Az általános célú operációs rendszerekkel szemben, a valós idejű rendszerek esetén a fókusz a pontos időzítésen és a determinisztikus működésen van. A fizikai térben történő eseményeknek pontos válaszidőt igényelhetnek, melyekre a rendszerünknek válaszolnia kell. Két alapvető ütemezési megoldás létezik:

- Kooperatív: A taszk váltás csak akkor következhet be, ha a feladat befejeződött vagy visszaadja a vezérlést az ütemezőnek
- Preemptív: A taszk váltás akkor is bekövetkezhet, ha van a jelenlegivel megegyező vagy nagyobb prioritású futásra kész taszk. Azonos prioritású taszkok round-robin időosztásos ütemezéssel kezelhetőek.

A FreeRTOS alapbeállításként a preemptív ütemezést használja. Ennek előnye, hogy egy magas prioritású taszk esetében a válaszidő legrosszabb estben a taszk váltási idő, míg kooperatív esetben a leghosszabb taszk ideje plusz a váltási idő.

A taszkok váltása és a megszakításkezelő rutinok során felléphet a közös erőforrások problémája. Ez azt jelenti, hogy egy adott erőforrást több taszk is használ és versenyhelyzet alakulhat ki, amely inkonzisztens memóriaterületet eredményezhet.

4.2.3 Szemaforok és mutexek

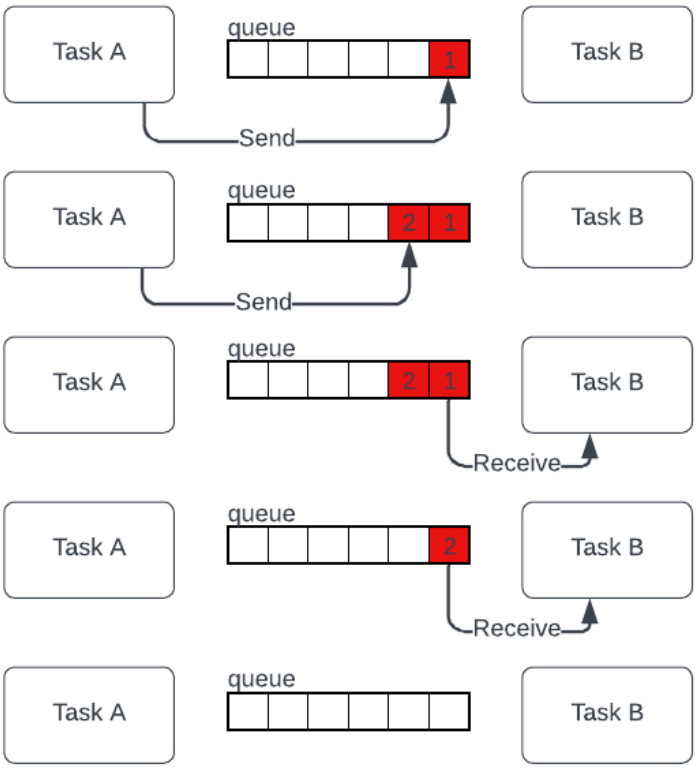
A versenyhelyzetetek elkerülése érdekében hozták létre a szemaforokat, melyek lehetnek binárisok vagy számláló típusúak. Segítségükkel megvalósítható a kölcsönös kizárás. A bináris szemafor szabad vagy foglalt állapotban lehet, jelezve az erőforrás hozzáférhetőségét. Ha egy taszk használni szeretné az erőforrást, de a szemafor foglalt állapotban van, akkor a taszk várakozó állapotba kerül, amíg az erőforrás szabaddá nem válik. Ezzel szemben a számláló szemaforok több értéket vehetnek fel, általában események számlálására, vagy az elérhető erőforrások mennyiségének nyomon követésére használják.

A mutexek olyan speciális bináris szemaforok, amelyek prioritás örökléssel vannak ellátva. Ez azt jelenti, hogy ha egy magas prioritású taszk várakozik egy mutexre, miközben azt egy alacsonyabb prioritású taszk foglalja, a kernel ideiglenesen megemeli az alacsonyabb prioritású taszk prioritását a magasabb szintjére. Ennek célja, hogy minimalizálja a magas prioritású taszkok várakozási idejét, ezáltal csökkenve a prioritás-inverzió hatását.

4.2.4 Queue

A sorok (queue) a taszkok közötti, valamint a megszakítási rutinok és taszkok közötti adattovábbítás elsődleges eszközei. A rendszerben minden kommunikációs forma alapja. A gyakorlatban többnyire FIFO szervezésű lista, amihez egy vagy több taszk is hozzáférhet. A sorok maguk gondoskodnak a kölcsönös kizárásról, ezért adott időben

legfeljebb egy taszk férhet hozzá a sorhoz. A sorba helyezéskor másolat készül az elemről, ez nagy méretű adatok esetén nem hatékony, ezért érdemes az adatra mutató pointert másolni. Ebben az esetben a kölcsönös kizárás nincs biztosítva, nekünk kell gondoskodnunk róla.



4.4. ábra Sorok működése

5 Tervezés

5.1 Rendszerterv

A rendszer megtervezése során három lehetséges megoldást vizsgáltam. A megoldások mérlegelése során a következő szempontokat mérlegeltem: a fejlesztés komplexitása és időigénye, a megoldás által nyújtott funkcionalitás, a jövőbeli továbbfejlesztési lehetőségek, valamint a megvalósítás költsége.

5.1.1 Egy mikrovezérlő alapú megoldás

Az első megoldásban a rendszer központi egysége egy nagy teljesítményű mikrovezérlő. A mikrovezérlő kezeli az érintő kijelzőn való megjelenítést, a szenzorok beolvasását és szabályozásokat. Ahhoz, hogy a vezérlő ennyi feladatot hatékonyan elláthasson, egy RTOS fut rajta. Továbbá a jövőbeli fejlesztések miatt fontos, hogy a rendszer Wi-Fi kapcsolattal rendelkezzen ezért egy ESP32-es vagy egy STM32H7-es mikrovezérlő merült fel megoldásnak. Ennek a konfigurációnak a legfőbb előnye, hogy rendkívül költséghatékony, illetve könnyedén találni olyan mikrokontroller modellt, ami elegendő perifériával rendelkezik a feladat megvalósításához. Azonban a 800x480 felbontású kijelző, több szabályozási kör és hálózatkezelés egyidejű futtatása olyan számítási és memória igénnyel rendelkeznek, amik a mikrovezérlők határait feszegetik. Emellett hátrány, hogy ebben az esetben nem áll a rendelkezésünkre általános célú operációs rendszer, így nincs lehetőségünk az általuk nyújtott, driverek és hatékony fejlesztési keretrendszerek használatára. További kompromisszum, hogy bár az utóbbi időben igen sokat fejlődtek a beágyazott felhasználói felület készítésére létrehozott könyvtárak, mint a TouchGFX és az LVGL, ezek továbbra is messze elmaradnak az általános célú operációs rendszereken elérhető keretrendszerek funkcionalitásától és praktikusságától.

5.1.2 Egykártyás számítógép alapú megoldás

Az utóbbi időben nagy népszerűségre tettek szert az egykártyás számítógépek (single-board computer továbbiakban SBC). Ezek olyan kisméretű számítógépek, melyeknek minden alapvető hardver komponense egyetlen nyáklapon található, mint a mikroprocesszor, ki és bemeneti interfészek, memória, egyetlen áramköri lapon található. Ezek a modulok olyan mikroprocesszort tartalmaznak, melyek képesek általános célú

operációs rendszer futtatására. Ez jelentős rugalmasságot kínál a választott programozási nyelv és keretrendszer szempontjából. További előnyük, hogy a legtöbb esetben a Wi-Fi és a hálózatkezelés sokkal szorosabban integrált ezekben a rendszerekben. Ennek a megoldásnak hátránya a magasabb ár, illetve kevésbé alkalmas valós idejű alkalmazásokra a futtatott operációs rendszerből adódóan, ami több szabályozási kör futtatásánál problémákat okozhat. Ezen felül jellemzően az ilyen eszközök sokkal kevesebb alacsony szintű perifériával rendelkeznek, mint az SPI és I^2C.

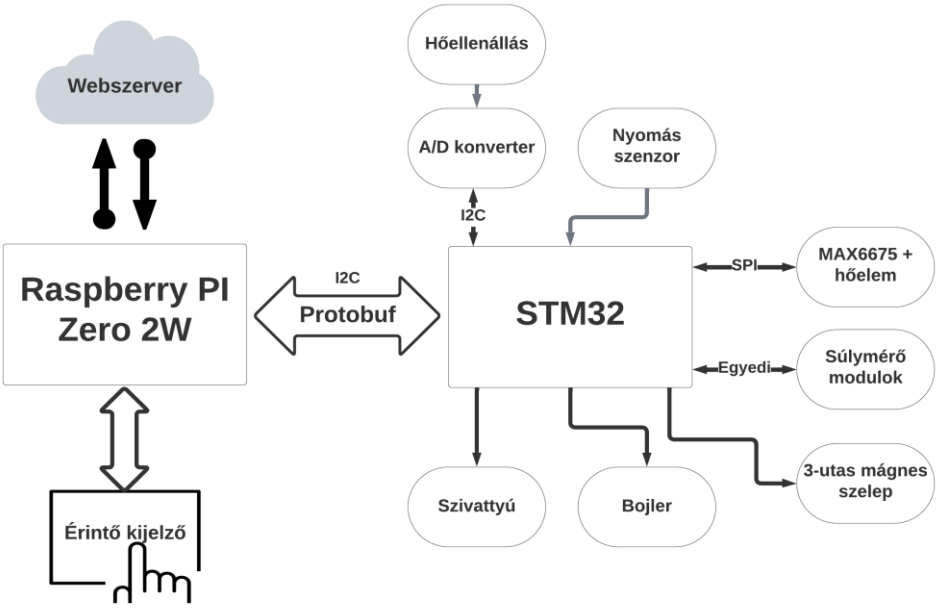
5.1.3 SBC és mikrovezérlő alapú megoldás

Ez a megoldás ötvözi az előző két megoldás előnyeit: az SBC végzi a megjelenítést, a felhasználói felület kezelését és a hálózatkezelést, míg a valós idejű feldolgozást igénylő feladatokat a mikrovezérlő hajtja végre. Ebben a konfigurációban a perifériák száma sem jelent problémát és a jövőbeli fejlesztésekre is nagyobb rugalmasságot biztosít. A megoldás egyik hátránya az ár, ez elsősorban a mikrovezérlős megoldással szemben releváns, mivel a mikrovezérlő költsége az SBC-hez képest csekély, különösképpen, mert használhatunk egy jóval kisebb teljesítményű mikrovezérlőt. Ezzel a megoldással azonban növeljük a rendszer komplexitását, hiszen két eszközre osztjuk fel feladatokat, amelyek egyes esetekben egymásra is épülhetnek. Ezért szükség van egy hatékony kommunikációs protokollra, ami, lehetővé teszi az adatcserét.

A fentiek alapján, a konfiguráció rugalmassága és bővíthetősége miatt ezt a megoldást választottam.

5.1.4 A rendszer blokkvázata

A fő komponensek kiválasztása után elkészítettem a rendszer blokkvázlát. Az ábrán láthatóak a rendszert felépítő elemek és a köztük lévő kommunikáció módja. SBC-nek egy Raspberry Pi Zero 2W-t választottam, kedvező ára és jó teljesítménye mellett rendkívül széles fejlesztői közösséggel rendelkezik. Mikrokontrollernek egy STM32F411-et választottam, amely ARM Cortex M4 maggal rendelkezik, így beépített lebegőpontos egysége (FPU) révén egyszerűsíti a fejlesztést, mivel a gyors számítások érdekében nem kell fixpontos aritmetikát használnunk. Emellett gazdag perifériakészlettel és elegendő ki- és bemenettel rendelkezik.



5.1. ábra Rendszer blokkvázlata

5.2 Kommunikáció a Raspberry Pi és az STM32 között

A Raspberry Pi és az STM32 közötti kommunikációhoz I^2C (lásd 3.1 fejezet) protokollt használok, mert nincs szükség nagy mennyiségű adat átvitelére. Továbbá előnyös, hogy a potokoll gondoskodik üzenet megerősítéséről (ACK) is, így biztosak lehetünk abban, hogy az üzenet megérkezett.

Ahhoz, hogy a két eszköz között adatstruktúrákat cseréljünk először szerializálni kell azokat, amely struktúra bájtfolyammá alakítását jelenti. A leggyakoribb

szerializációs módszerek közé tartozik a JSON és az XML, amelyek széles körben használatosak. Ezek a formátumok emberek számára is jól olvashatóak, és egyszerűen kezelhetőek. Viszont hátrányuk, hogy jelentős többlet adatot tartalmaznak és deszerializálásuk erőforrás igényes.

5.2.1 Protocol buffer [13]

A JSON és XML problémáinak megoldására fejlesztette ki a Google a protocol buffer röviden protobuf mechanizmusát, amely gyorsabb és csak minimális többletadatot alkalmaz.

Ahhoz, hogy adatokat küldjünk, az adatstruktúrát definiálnunk kell a Protobuf saját interfész leíró nyelvében (interface description language), amely *.proto* formátummal rendelkezik.

```
syntax = "proto2";

message ExampleMessage {
  optional string name= 1;
  optional int32 age = 2;
  repeated int32 grades =3;
}
```

1. kódrészlet Példa Protobuf leíró fájl

A fent látható kódrészlet egy példa Protobuf üzenetet mutat be. Az első sorban meg kell adnunk, hogy a Protobuf melyik verzióját kívánjuk használni. Minden mezőhöz meg kell adnunk egy azonosító számot, amelynek egyedinek kell lennie az üzeneten belül. Ezen kívül minden mezőhöz egy jelzőt kell rendelnünk az alábbiak közül:

- optional: Az üzenetnek opcionálisan tartalmazhatja, ha nem állítjuk be az értékét akkor alapértelmezett értéket kap.
- required: Az üzenetnek kötelezően tartalmaznia kell pontosan egy ilyen mezőt. Ennek használata nem ajánlott, mivel számos problémát okozott, ezért a proto3 verzióból ezt a lehetőséget kivették.
- repeated: Az üzenet tetszőleges számú ilyen mezőt tartalmazhat. Az ismétlődő értékek sorrendje megmarad.

Egy fájlban akár több üzenet leíró struktúrát definiálhatunk és üzeneteket egymásba ágyazhatunk.

A Google Protobuf C++ nyelven íródott, és natívan támogatja a leggyakrabban használt objektumorientált programozási nyelveket. A projektemben azonban a mikrokontroller kódját C nyelven készítettem, amelyhez nincs hivatalos Protobuf implementáció. Erre nyújt megoldást a *nanopb* könyvtár, amely kifejezetten 32 bites mikrokontrollerekhez fejlesztett, ANSI-C alapú Protobuf implementáció.

5.2.2 A rendszer üzenetei

A kávéfőző különböző működési állapotaiban az eszközök különböző üzenet struktúrákat váltanak egymás között. Mivel a két eszköz I^2C protokoll segítségével kommunikál, ezért mindig a Master kezdeményezheti az üzenet küldését és fogadását. A rendszerben a Raspberry Pi a Master. A protobuf üzenetek nem tartalmaznak beépített információt az üzenet kezdetéről és végéről, ezért a minden üzenet küldése előtt az első byte-ban az üzenet hosszát továbbítom.

A rendszer fő üzenet struktúráját a Raspberry Pi küldi a mikrovezérlőnek. Az üzenetben továbbítja a kívánt főzési hőmérséklet, nyomást és kávé mennyiséget. Illetve ez a struktúra jelzi, a fő állapotok közötti váltást. A protobuf lehetővé teszi az üzenetekbe enum változók ágyazását, amely ideális az aktuális állapot továbbítására. Az üzenet tartalmát a felhasználói felületen beállított értékek és folyamatok határozzák meg.

```
State {
    BOOTING = 1;
    HEATING = 2;
    HEATING_FOR_STEAM = 3;
    READY = 4;
    BREW = 5;
    STEAM = 6;
    ERROR = 7;
}
message SystemSettings {
    optional State state = 1;
    optional float targetTemperature = 2;
    optional float targetPressure = 3;
    optional float targetWeigth = 4;
}
```

2. kódrészlet Paramétereket beállító stuktúra

Amikor a kávéfőző készenléti álláspontban van 0.5 másodpercenként frissítést kér a mikrovezérlőtől. A válasz tartalmazza az aktuális hőmérsékletet, a mérleg által mért

értéket, illetve a hibakódot. Ezeket az értékeket meg kell jeleníteni a kijelzőn, ezért folyamatosan frissítenünk kell őket.

```
message IdleStatusUpdate {
    optional float currentTemperature = 1;
    optional float currentWeight = 2;
    optional uint32 errorCode = 3;
}
```

3. kódrészlet Állapotjelző struktúra készenléti állapotban

Ha a RPi által küldött állapot „BREW” módba vált megkezdődik a főzés. Annak érdekében, hogy a grafikonok kirajzolása folyamatosabb legyen ekkor a főzési paramétereket 0,05 másodperceként, azaz ~20 Hz-es gyakorisággal kéri le. Ebben az esetben a nyomást is továbbítja a vezérlő. A főzés azonnal befejeződik, ha az állapot bármilyen más állapotra vált.

```
message BrewStatusUpdate {
    optional float currentTemperature = 1;
    optional float currentPressure = 1;
    optional float currentWeight = 2;
    optional uint32 errorCode = 3;
}
```

4. kódrészlet Állapotjelző struktúra főzés közben

5.2.3 Üzenet kódolás

Ahhoz, hogy a kívánt üzeneteket el tudjuk küldeni, először a *.proto* fájlokból kódot kell generálnunk. A Raspberry Pi-ra C++ nyelven a protobuf compiler segítségével fordítjuk le a fájlokat, míg a mikrokontrollerre C nyelven a nanopb-vel. A fordítások eredményeként egy *source* és egy *header* fájlt kapunk a fordított nyelven. A kapott fájlokat, valamint a protobuf függőségeket hozzáadjuk a megfelelő projekthez. Ezt követően kódolhatunk és dekódolhatunk üzeneteket. Először létre kell hoznunk az üzenet struktúráját, majd az egyes elemekhez értékeket kell rendelnünk. Ezután az adat struktúrát szerializáljuk a SerializeToOstream() függvény segítségével.

```
    SystemSettings settings;
    settings.set_state(SystemSettings::HEATING);
    settings.set_targettemperature(95.0f);
    settings.set_targetpressure(9f);

    std::ofstream output("settings.bin", std::ios::binary);
```

```
if (!settings.SerializeToOstream(&output)) {
        std::cerr << "Failed to write system settings." << std::endl;
        return -1;
}
```

5. kódrészlet Protobuf üzenet kódolása

6 Hardver tervezés

A következő feladat a nyomtatott áramköri lap tervezése volt, melynek elsődleges célja az, hogy megbízható tápellátást biztosítson a központi feldolgozó egységeknek, és zajmentes összeköttetést alakítson ki, amely lehetővé teszi a zavartalan adatátvitelt a további modulokkal és szenzorokkal.

6.1 Kapcsolási rajz

A kapcsolási rajzot a KiCad-el készítettem, ami egy ingyenes nyílt forráskódú tervező progam.

Mivel az eszköz a 230V-os hálózathoz csatlakozik, ezért egy kereskedelmi forgalomban kapható, hitelesített tápegység mellett döntöttem. A Raspberry Pi Zero 2W adatlapja szerint az eszköz 5V-os 2,5A-es tápra van szüksége, illetve a többi eszközt is táplánunk kell. További fontos szempont volt, hogy a tápegység rendelkezzen túlfeszültség és rövidzár védelemmel. Ezek alapján végül egy IRM 15-5 modult választottam, amely 5V-os kimenettel rendelkezik és maximális kimeneti árama 3A.

A mikrokontroller működéséhez 3.3V-os feszültségre van szükség. A helytakarékosság és praktikusság miatt egy LDO-t (Low-dropout regulator) választottam. Ezek hátránya, hogy a bemeneti és kimeneti feszültség különbségéből származó energiát hő formájában disszipálják. Ebben az esetben a tápegység hatásfoka megközelítőleg 66%. Erre egy TPS8B8633-as modult választottam, ami 500 mA maximális kimeneti árammal bőven elegendő lesz a mikrovezérlő és a többi 3.3V-os IC táplálására.

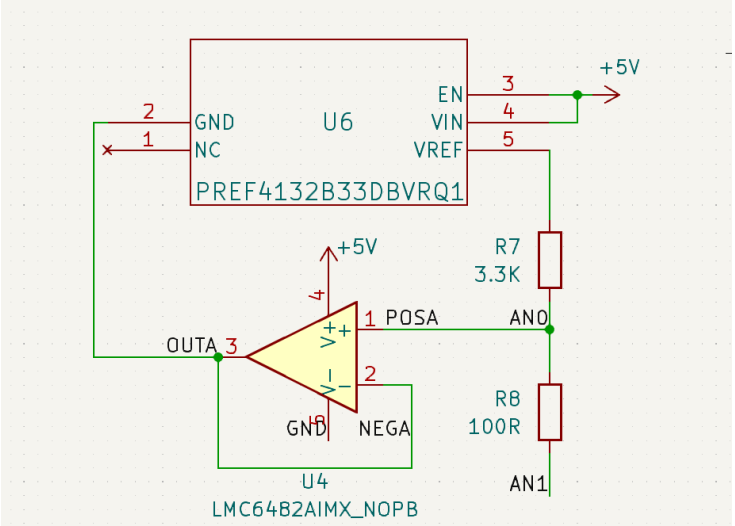
6.1.1 Hőmérséklet mérés

A 4.1.4 -es fejezetben részleteztem a 4-vezetékes hőellenállásos mérés működését. Ebben a fejezetben ennek áramköri megvalósításáról lesz szó.

A mérés megvalósításához négy fő komponensre van szükség: egy 1mA-es precíziós áramforrásra, egy A/D átalakítóra és magára a 4-vezetékes PT100-as hőellenállásra és a referencia ellenállásra.

A precíziós áramforrás kapcsolási rajza az alábbi ábrán látható. A kapcsolás egy mérőerősítőt és egy precíziós feszültségreferenciát tartalmaz. A követő erősítő mindig a terhelésen eső feszültséget vezeti a feszültség referencia föld csatlakozására, így a

kimenethez képest állandóan 3,3 V-ot biztosít. Ezáltal garantált, hogy az R7-es ellenálláson 3,3 V feszültség esik. Ennek alapján a kimenő áram könnyen kiszámítható az $I_{ki} = \frac{3{,}3}{R_7}$ összefüggés segítségével. Az 1 mA-es áramforrás eléréséhez egy precíziós 3,3 kΩ-os ellenállást alkalmaztam. A mérőerősítő kiválasztásánál kiemelten fontos szempont, hogy rail-to-rail típusú legyen. Ez azt jelenti, hogy a kimeneti feszültség tartománya közel van a tápfeszültség alsó és felső határához. Ez azért lényeges, mert a PT100-as érzékelőn és a referenciaellenálláson kis feszültség esik. Emiatt a mérőerősítőnek képesnek kell lennie alacsony, a tápfeszültség földszintjéhez közeli kimeneti feszültség előállítására.



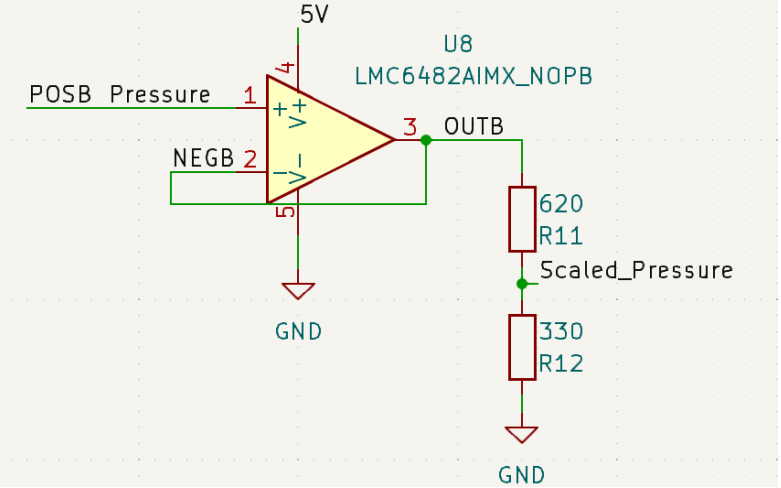
6.1. ábra Precíziós áramforrás kapcsolási rajz

Az AD-átalakító kiválasztásánál az ADS1115-re esett a választásom, ám ez később hibás döntésnek bizonyult, mivel a modul nem teszi lehetővé külső feszültségreferencia megadását. Emiatt a 4.1.1-es fejezetben ismertetett mérési összeállítás ezzel az eszközzel nem megvalósítható. Ugyanakkor az ADS1115 képes két differenciális bemenet kezelésére, ami lehetőséget adott a probléma szoftveres megoldására: a referencia- és a PT100-ellenállás értékeit felváltva olvasom ki, majd ezek alapján számítom ki a relatív változást.

6.1.2 Nyomás mérés

A 3.2.2-es fejezetben részletesen bemutattam a nyomásszenzorok működését, valamint az alkalmazáshoz végül kiválasztott szenzort. A XDB401-es szenzor 0,5 – 4,5 V-os kimenettel rendelkezik, azonban a mikrokontroller beépített konvertere legfeljebb 3,3V-os bemeneti feszültséget bír el. Ennek megoldására két lehetőség merült fel: egy

külső A/D konverter használata, vagy a 4,5 V skálázása 3,3V-ra és a beépített konverter használata Mivel ebben az esetben nincs szükség differenciális mérésre, és a beépített konverter is megfelelő felbontást biztosít, az utóbbi megoldást választottam. A kis impedanciájú kimenet biztosítása érdekében egy követő erősítőt építettem be. Ezután egy ellenállás osztó segítségével skáláztam a nyomásszenzorból érkező feszültséget. Bár a szenzor kimeneti feszültsége maximálisan 4,5 V, figyelembe véve, hogy az erősítő 5 V-os tápfeszültséget kap, az ellenállásokat úgy választottam meg, hogy ha a kimeneti feszültség mégis elérné az 5 V-ot, akkor se károsodjon a mikrokontroller A/D bemenete. Ezzel ugyan veszítünk a rendszer felbontásából, de a felbontás továbbra is bőven elegendő marad a kívánt mérési tartományhoz. Ez azt jelenti, hogy $\frac{3,3}{5}$ -ös azaz 0,66-os skálázásra van szükség. Erre a célra egy 620 Ω és egy 330 Ω -os 1%-os toleranciájú ellenállást választottam. Ezzel a választott ellenállás-párossal a következő arányt kapjuk: $\frac{620}{620+330} = 0.6526$.



6.2. ábra Nyomás mérés kapcsolási rajz

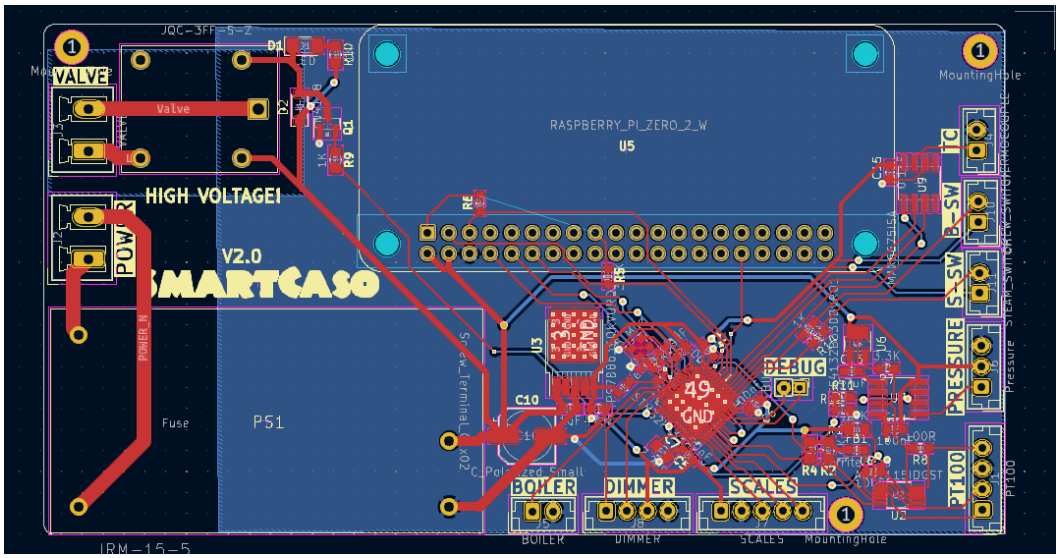
A mikrokontroller beépített ADC-je 12 bites felbontású. A szenzor kimeneti feszültsége 0,5–4,5 V, amely a 0–12 bar tartománynak felel meg. A skálázás után a kimeneti tartomány 0,3263 V és 2,9367 V között lesz. Így végül a felbontás $\frac{12\ bar}{\frac{2,6104}{3,3}\ 2^{12}} = 0,0037\ bar$, ez teljes mértékben megfelel az előzetes követelményeknek és meghaladja a szenzor által nyújtott 1%-os pontosságot.

6.2 Huzalozási terv

A NYÁK megtervezése során az egyik korlátozó tényező a kávéfőzőben rendelkezésre álló korlátozott hely volt. Ez számos nehézséget okozott a tervezés során és ez volt az egyik fő oka, hogy végül újratervezésre volt szükség.

A kapcsolási rajz elkészítése után a huzalozási terv elkészítése következett. Mivel a hardveren nincs nagy sebességű kommunikációt igénylő komponens, így az áthallás (Cross-talk) és az elektromágneses interferencia (EMI) kis eséllyel okoznak problémát. Ezért a tervezés során egy két rétegű áramköri lap használata mellett döntöttem.

A felső oldalon található az összes komponens és a jelvezetékek túlnyomó része, míg a második réteget földkitöltés borítja. A tervezés során arra törekedtem, hogy a jelvezetékek lehetőség szerint a felső rétegen maradjanak, és csak akkor kerüljenek az alsó rétegre, ha másképp nem megoldható. Ez különösen fontos magasabb frekvenciák esetén, mivel a visszatérő áramok az adott vezetősáv alatt, a legkisebb impedanciájú útvonalon térnek vissza. Emiatt lényeges, hogy a földkitöltés a lehető legnagyobb mértékben sértetlen maradjon, minimalizálva az elektromágneses zavarok kialakulását. A bal oldalon találhatóak a nagy feszültségű komponensek, míg a jobb oldalon a kis feszültségűek helyezkednek el. Ennek megfelelően biztonsági szempontból a bal oldalon nincs földkitöltés.

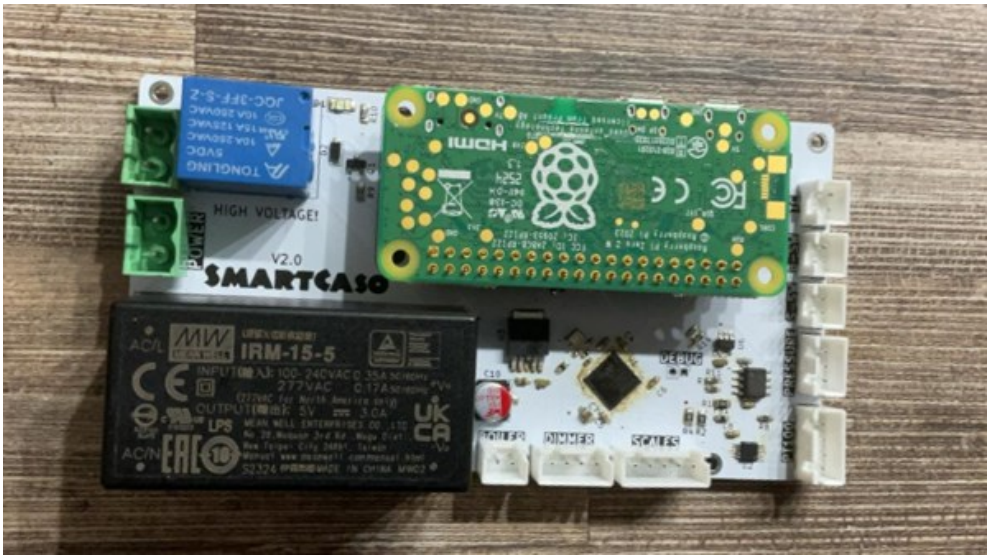


6.3. ábra Huzalozási terv

6.3 Gyártás

A nyomtatott áramköri terv készítése során az egyik legfontosabb szempont, hogy olyan tervet készítsünk, amely a gyártás során is megvalósítható. A NYÁK gyártók gyakran közzétesznek olyan dokumentumokat, amelyek tartalmazzák a minimális gyártási tűréseket és képességeket. Ezt a dokumentumot figyelembe kell vennünk a tervezés során, mert a tervünket gyártás előtt DRC (Design Rule Check) ellenőrzésnek vetik alá. Ha a tervezett áramkör nem felel meg a gyártási szabványoknak, akkor a szűkebb tűrések miatt többletköltséget számolnak fel vagy elutasítják a gyártást. A technológiai paramétereket a legtöbb tervező szoftverbe betáplálhatjuk, így automatikusan jelzi, ha a terv megszegi a gyártási szabályokat.

Az áramköröm gyártását a JLCPCB céggel végeztettem el. A NYÁK mellé stencil lapot is rendeltem, így a forrasztást könnyedén el tudtam végezni reflow forrasztással.

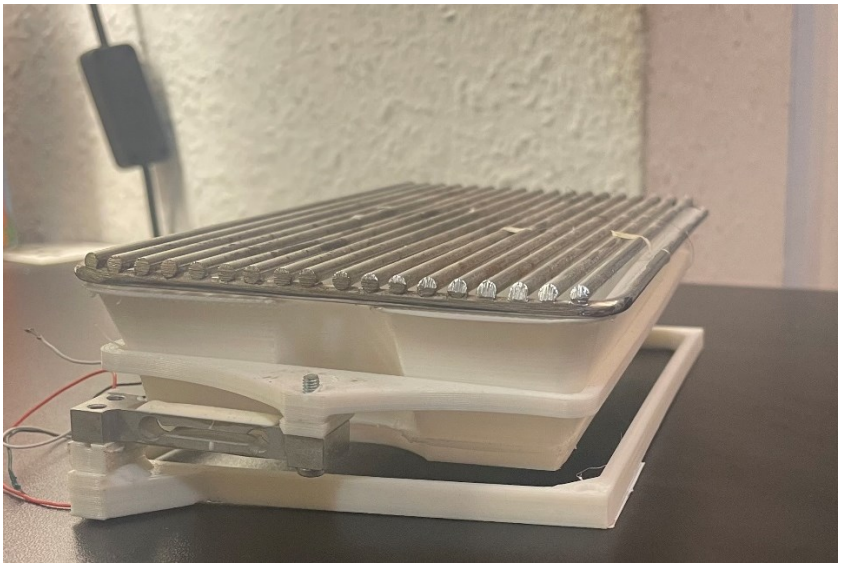


6.4. ábra Összeszerelt NYÁK

6.4 Beépített mérleg

A beépített mérleg megvalósításához először egy merev alapot kellett terveznem, amelyre az erőmérő cellák rögzíthetők, valamint egy platformot, amelybe a csepegtető tálca helyezhető. A pontos méréshez fontos, hogy az alap megfelelő merevséget biztosítson. A mérleghez két darab 500g-os egy-pontos cellát választottam. Ez azonban később hibás döntésnek bizonyult, mivel a cellák kis érintkezési felülete és vékony kialakítása nem nyújtott megfelelő stabilitást a platform számára. Ez azt eredményezte, hogy a csepegtető tálca pereme a kávéfőző házára támaszkodott, ami ellehetetlenítette a

mérést. A problémák ideiglenes áthidalására az eredeti fém csepeggető tálcát lecseréltem egy 3D nyomtatottra. Ez jelentősen csökkentette a cellák alapterhelését és javított a problémán, de nagyobb terhelések esetén továbbra is fennáll. Jövőbeni fejlesztési tervek között szerepel a jelenlegi erőmérő cellák lecserélése nagyobb mérési kapacitású, stabilabb változatokra.



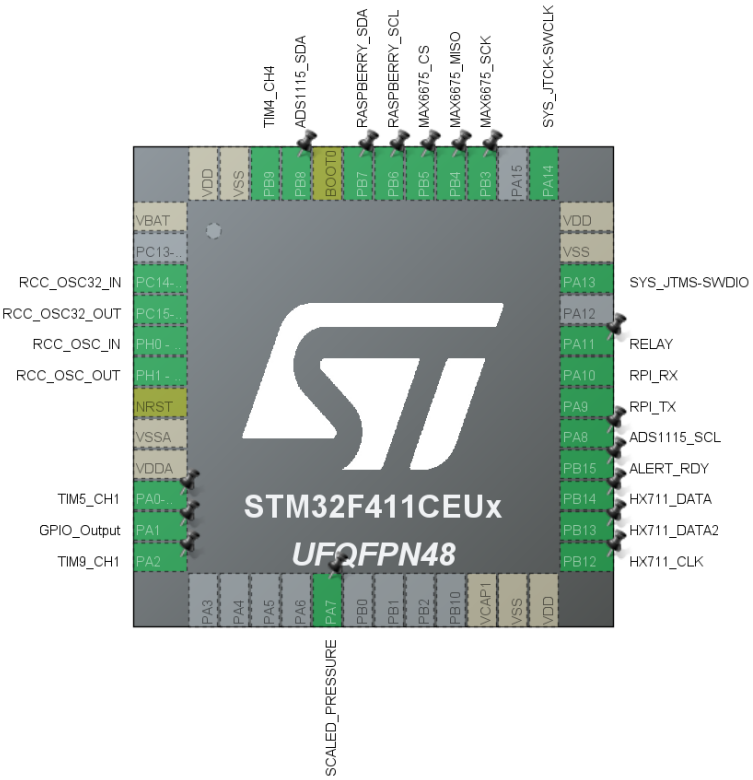
6.5. ábra Beépített mérleg

7 Szoftverfejlesztés

7.1 Mikrovezérlőn futó program

7.1.1 STM32CubeIDE

A mikrovezérlőn futó programot, az ST által fejlesztett STM32CubeIDE használatával készítettem Ez egy kifejezetten az STM32 mikrovezérlő családhoz tervezett, Eclipse-alapú integrált fejlesztőkörnyezet. Az egyik legnagyobb előnye, hogy tartalmazza az STM32CubeMX konfigurációs eszközt, amely egy intuitív, grafikus felületet kínál a perifériák, órajelek és egyéb rendszerbeállítások konfigurálásához. Ennek segítségével automatikusan legenerálhatjuk a mikrovezérlőt inicializáló C kódot. További előnye, hogy támogatja az ST-Link debug interfészt, amely lehetővé teszi, hogy a mikrovezérlőn futó kódot valós időben, közvetlenül a hardveren debuggoljuk, mintha az a saját gépükön futna.



7.1. ábra Grafikus konfigurációs felület

7.1.2 STM32 HAL

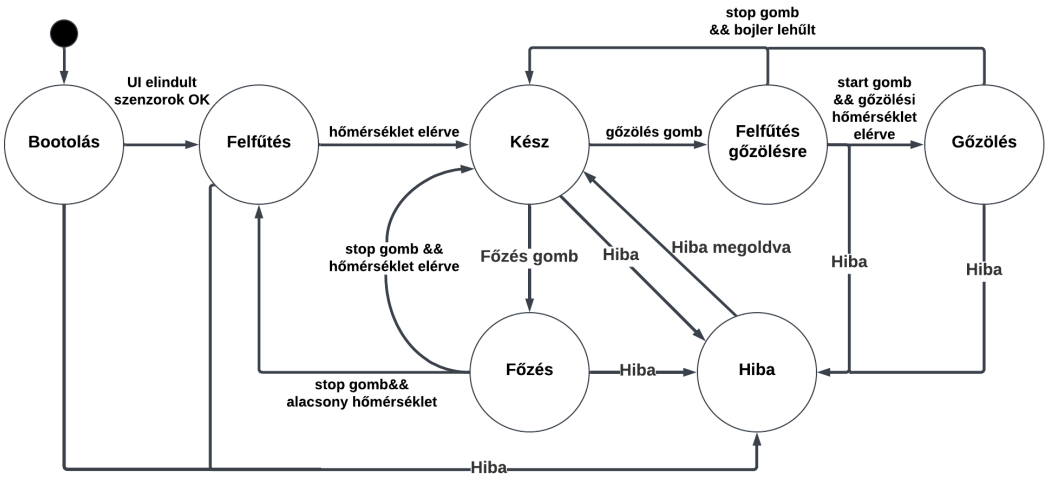
Az ST ezen felül biztosítja az STM32 HAL (Hardware Abstraction Layer) könyvtárat, amely egy olyan hardverabsztrakciós szoftverréteg, amely lehetővé teszi a mikrokontroller perifériáinak egyszerű kezelését magas szintű API (Application Programming Interface) segítségével. A könyvtár két fő csoportra van oszlik: generikus driverekre, melyek minden STM32 mikrovezérlőre azonosak és az extension driverekre, ezek specifikus modellekre, sorozatokra vonatkoznak. Ennek köszönhetően nem szükséges közvetlenül a regiszterek kezelése, ezzel növelhetjük programunk hordozhatóságát. De fontos megemlíteni, hogy ez a portabilitás teljesítmény veszteséggel jár az alacsony szintű könyvtárak alkalmazásával szemben.

A mikrovezérlőn FreeRTOS operációs rendszer, melynek főbb funkcióit és működését a 4.2-es fejezetben fejtettem ki. A program 4 különböző taszkból áll:

- Fő állapotgépet kezelő taszk
- Hőmérséklet-szabályozás taszk
- Nyomásszabályozás taszk
- Súlymérés taszk

7.1.3 A rendszer állapotgép

Az állapotgép kezeléséért felelős taszk. A gép működése 7 különböző fő állapotra bontható, minden állapotváltás a Raspberry Pi utasítására hajtódik végre. Ez alól kivételt képez a hiba állapot, amelybe a mikrovezérlő magától is képes belépni.



.

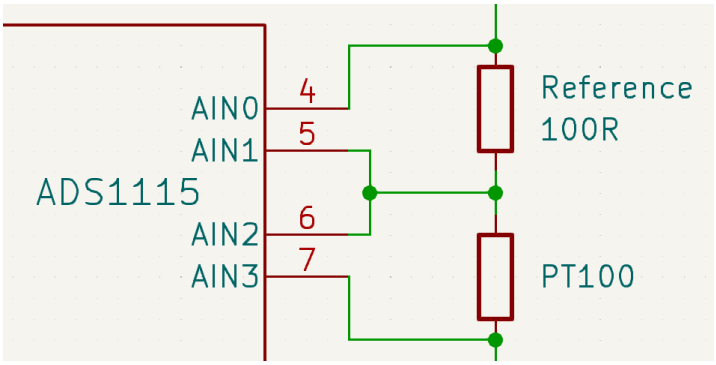
7.2. ábra A rendszer állapotgépe

7.1.4 Hőmérséklet-szabályozás

Az ADS1115 modul egy 16 bites delta-sigma (ΔΣ) elven működő analóg-digitális átalakító, amely I^2C interfésszel rendelkezik. Az eszköz 4 analóg bemenetet biztosít, melyeket használhatunk egy kimenetű (single-ended) jelek mérésére vagy két különböző differenciális mérésre. Emellett rendelkezik egy beépített erősítővel, illetve egy komparátorral, amely segítségével külső megszakításokat küldhetünk.

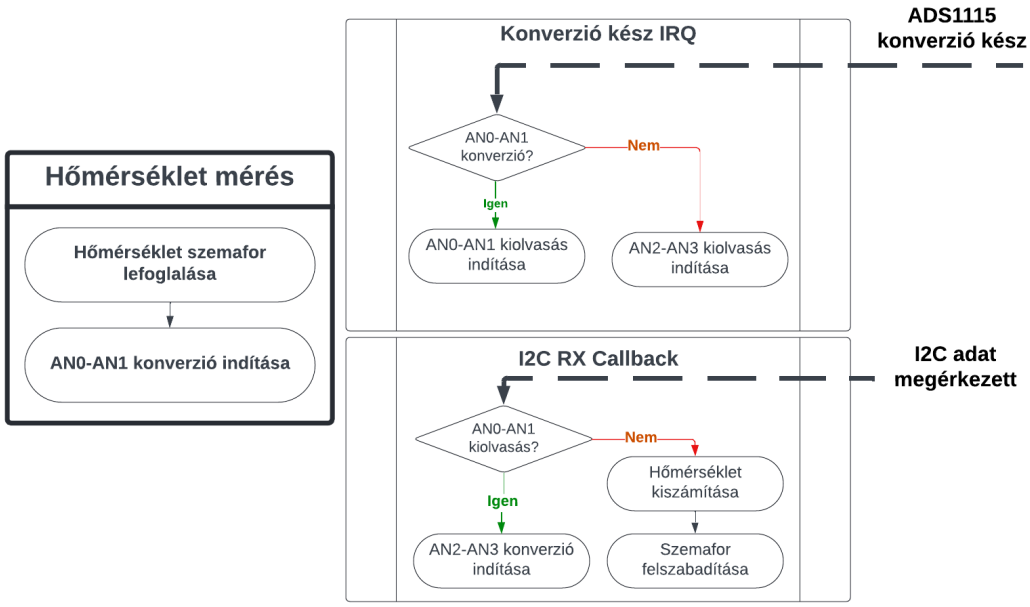
A modul négy regiszterrel rendelkezik: egy konverziós regiszterrel, egy konfigurációs regiszterrel és egy-egy regiszterrel a komparátor alacsony és felső határai beállításra.

A konfigurációs regiszter segítségével megadhatjuk, mely bemeneten vagy differenciális mérés esetén bemenetek között szeretnénk mérni, konfigurálhatjuk a beépített erősítés mértékét, a mintavételezési időt és a komparátor működési módját [15].



7.3. ábra A/D konverter bemeneteinek bekötése

Az összeállításban két differenciális mérést végzünk, az egyik méréssel a precíziós referencia ellenálláson eső feszültséget mérjük, a másik méréssel a PT100 hőellenálláson eső feszültséget. Az eszköz egyszerre egy mérést tud végezni, ezért a méréseket felváltva kell elvégezni. Erre jó megoldást nyújt az ADS1115 „Alert ready” funkciója, amelyet a komparátor, megfelelő beállításával kapcsolhatunk be. A modul a konverzió végeztével a láb beállítása szerint felfutó vagy lefutó éllel jelzi a mérés elkészültét. A hőmérséklet kiolvasását nem blokkoló megszakításos függvényekkel végzem, ez azt jelenti, hogy amíg az adat küldése és fogadása folyamatban van a processzor nem „pollingolja” az I^2C perifériát. Az adatátvitel végeztével a periféria létrehoz, egy megszakítást, amelyben kezeljük az eseményt. Ezzel biztosítjuk, hogy minél kevesebb ideig blokkoljuk a processzort.



7.4. ábra Hőmérséklet mérés folyamat ábrája

Először lefoglaljuk a hőmérséklet adatok memóriaterületét védő szemafort. Ez azért szükséges, hogy az új hőmérséklet adatok kiolvasása közben a szabályozó és a protobuf taszkok ne férjenek hozzá ezekhez az adatokhoz. Ezt követően az ADS1115 konfigurációs regiszterének írásával elindítjuk a AN0-AN1-csatorna konverzióját. A program mindaddig tovább fut amíg a rendszer nem kapja meg a konverzió kész megszakítást az AD modultól.

Mivel a felváltva olvassuk ki a referenciát és a PT100 ellenállást, meg kell vizsgálnunk melyik csatorna konverzióját indítottuk el. Attól függően, hogy AN0-AN1 vagy AN2-AN3-csatornát olvastuk a hőmérséklet tömb megfelelő helyére kiolvassuk a konverziós regisztert. Ha a kiolvasás sikeres volt az I^2C küld egy HAL_I2C_MasterRxCpltCallback() megszakítást. Ekkor megvizsgáljuk, hogy melyik érték kiolvasását indítottuk el legutóbb, ha AN0-AN1-csatornát akkor elindítjuk az AN2-AN3 konverzióját, ha és a folyamat megismétlődik a másik ágon. Ha a AN2-AN3 kiolvasását indítottuk el akkor a kiolvasást befejeztük és kiszámítjuk a hőmérséklet értéket:

$$R_{PT100} = \frac{U_{PT100}}{\frac{U_{ref}}{R_{ref}}}$$

Az ellenállás hőmérőknél ismertetett összefüggés az ellenállásra:

$$R_{PT100} = 100 \cdot [1 + 3.9083 \cdot 10^{-3} \cdot T - 5.775 \cdot 10^{-7} \cdot T^2]$$

A hőmérsékletet megkapjuk a másodfokú egyenlet megoldásával T-re.

A kávéfőző különböző állapotaihoz, a bojler szabályozását az adott állapothoz igazított karakterisztikával kell végezni. Az optimális szabályozás érdekében a következő üzemállapotokat különböztethetjük meg: felfűtés, hőmérsékleten tartás és kávéfőzés. A szabályozás megvalósítására egy PID szabályozót választottam.

A felfűtés során kiemelt figyelmet kell fordítani arra, hogy a szabályozás elkerülje a túllövést, mivel a bojler hőmérséklete lassan csökken, ugyanakkor törekedni kell a lehető legyorsabb felfűtésre. Kávéfőzés közben azonban a tartályból hideg víz áramlik a bojlerbe, ezért ebben az üzemmódban dinamikusabb szabályozásra van szükség a lehűlés elkerülése érdekében. A PID paraméterek meghatározásához több módszert is kipróbáltam: a MATLAB PID tuner eszközét, a bojlert kapacitásként modelleztem és pólus-zérus kiejtést végeztem. De ezek a megközelítések nem vezettek kielégítő eredményre, így végül empirikus úton állapítottam meg a paramétereket.

A szabályozási algoritmus alapján számolt rendelkező jelből egy PWM impulzus szélességet számolok. A PWM jel egy TRU COMPONENTS szilárd test relét vezérel. A relé nyitott állapotban rákapcsolja a 230V-os feszültséget a fűtőszálra.

A szilárdtest relé zero-crossing üzemmódban működik, ez azt jelenti, hogy a már vezető relé csak akkor nyeri vissza szigetelőképességét, ha a rajta átfolyó áram zérusra csökken. Mivel a hálózati feszültség 50 Hz-es ezért nem lehet magas frekvenciájú PWM jelet használni, mert akkor a váltások két nullpont között történnek nincs hatásuk a kimeneti jelre. A szabályozáshoz ezért 1 Hz-es PWM jelet választottam, ez nem okoz problémát, mert a szakasz kellően lassú. Mivel 1 másodperc alatt 100-szor metszi a nulla pontot és a működés miatti bekövetkező hiba egy félperiódus, így a maximális hiba, amit ezzel a rendszerbe viszünk 1%.

7.1.5 Súlymérés

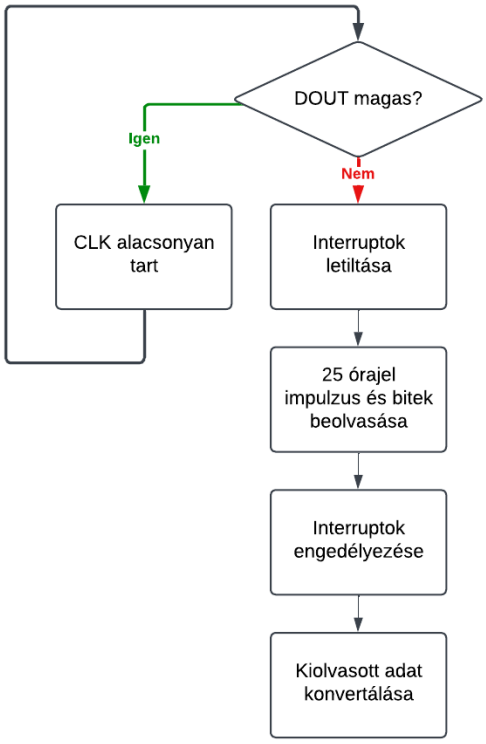
A beépített mérleg mechanikai megvalósítását a 6.4-es fejezetben részleteztem. A két erőmérő cella által kis feszültségváltozásainak méréséhez nagy felbontású analóg-digitális konverterekre van szükség, amelyek beépített erősítővel rendelkeznek Erre a két darab HX711 24 bit-es AD konvertert választottam. A HX711 modul kétvezetékes, egyedi soros kommunikációt használ, és rendelkezik egy CLK és egy DOUT pinnekkel. Az eszköz két csatornával rendelkezik, a csatornák közötti választás, illetve a csatornák erősítésének beállítása a kiküldött órajel impulzusok számával történik. Azért használtam

két modult mert a két csatorna különböző erősítéssel rendelkezik, ez szoftveresen áthidalható lenne, de a modulokat együtt adták a cellákkal, így ez kézenfekvőbb volt. A projekt során az 'A'-csatornát használtam 128-as erősítéssel.

PD_SCK Pulses	Input channel	Gain
25	A	128
26	B	32
27	A	64

7.5. ábra HX711 csatornái és erősítései [16]

Amikor a modul nem áll készen az adatok küldésére abban az esetben a DOUT lábat logikai magas értéken tartja, ekkor az órajelet alacsonyan kell tartanunk. Amikor DOUT alacsonyra esik, az eszköz készen áll az adatok küldésére, ekkor 25 pozitív óraimpulzussal kiolvassuk az adatokat. Minden egyes órajel impulzus egy bitet tol ki, az MSB-től kezdve, amíg mind a 24 bit kiküldésre kerül. A 25. impulzusra a modul DOUT-ot magasra húzza vissza. Fontos, hogy az adatok kiolvasása során ne érkezhessen interrupt, mert ha a CLK pin alacsonyról magasra változik és $60\mu s$-ig magas marad, akkor a modul alvó állapotba kerül.



7.6. ábra HX711 kiolvasása

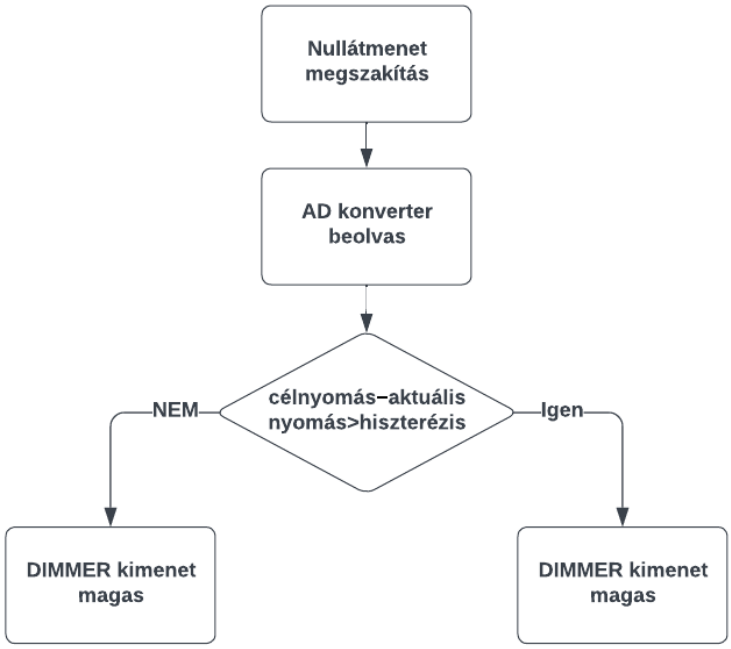
7.1.6 Nyomásszabályozás

A 2.2-es fejezetben bemutattam, hogy a vibrációs szivattyú 50 Hz-es frekvenciájú impulzusokkal hozza létre a nyomást. Ahhoz, hogy a későbbiekben pontos térfogatáram számításokat tudjunk végezni, elengedhetetlen, hogy pontosan tudjuk a dugattyú visszahúzódásainak a számát. Ezért a hőmérséklet-szabályozáshoz hasonló zero-cross relé nem alkalmas erre a feladatra. Ennek oka, ha a hálózati feszültség és a szabályozó PWM jelünk nincs szinkronban, előfordulhat, hogy a vezérlőjel pont a nullátmenet után aktiválódik, így szivattyú nem kap feszültséget. Ellenkező esetben, ha jel nullára vált közvetlenül a keresztezés után, a szivattyú feszültséget kap, nemkívánatos impulzust okozva.

A feladatra egy olyan relé sem alkalmas, amely bármikor képes kapcsolni, de nem rendelkezik nullátmenet érzékeléssel, mert a szivattyú feszültség szinusz hullámát nem vághatjuk el bárhol. Ha a kitöltési tényező 50% alá csökken a dugattyú nem tud megfelelően visszahúzódni, ezért nem képes megfelelő nyomás előállítani. Ezért a szabályozáshoz olyan eszközre van szükség, amely érzékelni képes a nullpontátmeneteket.

A használt feszültség vezérlő modulnak a két logikai csatlakozója van. Egy „DIMMER” bemenet, amelyre, ha logikai 1-es jelet adunk a modul kinyit és átengedi a feszültséget. És egy „ZC” kimenet, amely minden nullátmenet esetén magasra vált. A mikrovezérlőn a nullpont-átmenetet jelző csatlakozót külső megszakításként kell konfigurálni. Ennek köszönhetően minden egyes nullátmenetkor egy megszakítási rutin indul, amely a szinuszjel továbbításának logikáját végzi.

A nyomásszabályozást a következő metódussal végzem: A feszültségszabályozó érzékeli a nullpontátmenetet, és ezt jelzi a mikrovezérlőnek. A mikrovezérlő az aktuális és a kívánt nyomás alapján dönt arról, hogy az adott fél szinusz hullámot továbbítja-e. Amennyiben a továbbítja, a feszültségszabályozó „DIMMER” bemenetére magas szintű jelet kapcsol. Ennek megvalósítására egy kétállású hiszterézises szabályozást implementáltam. A szabályozás során minden nullaátmenetnél megvizsgáljuk a kondíciót és amennyiben a kívánt tartomány alatt vagyunk átengedjük a fél szinusz hullámot, amennyiben a kívánt tartományt elértük a kapcsolót zárva tartjuk.



7.7. ábra Nyomásszabályozás folymat ábra

Ez a szabályozási módszer rendkívül egyszerű és minimális számítási igényekkel rendelkezik.

7.2 Raspberry Pi-on futó program

A Raspberry Pi-n futó programot a Qt könyvtár használatával fejlesztettem. A kódot lokálisan, a Windows Subsystem for Linux (WSL) környezetében írtam, a Visual Studio Code remote explorer használatával. Majd a projektet CMake build rendszer segítségével keresztfordítottam a Raspberry Pi-ra.

7.2.1 Qt [17]

A Qt egy nyílt forráskódú, platformfüggetlen fejlesztői keretrendszer. Elsősorban grafikus felhasználói felületekkel rendelkező programok készítésére fejlesztették, de számos más területen alkalmazható. A Qt könyvtár C++-ban van megírva és leggyakrabban a fejlesztés is ebben folyik, de az utóbbi időben már a Python-t is támogatja.

A Qt számos olyan funkciót kínál, amelyek közvetlenül nem kompatibilisek a C++ nyelvvel. Az egyik leghatékonyabb eszköz a signals and slots mechanizmus, amely lehetővé teszi az objektumok közötti szinkron és aszinkron kommunikációt. Minden QObject típusból leszármazó osztályban létrehozhatunk jeleket (signals), amely akkor

történik, amikor egy objektum állapota megváltozik vagy egy esemény következik be. Például egy gomb megnyomása, egy adatfrissítés vagy bármilyen más, az alkalmazás logikájában érdekes állapotváltozás egy jelet generálhat. A slotok lényegében normál C++ függvények, de a fő különbség, hogy kifejezetten a jelekhez vannak kapcsolva, így, ha egy jel aktiválódik, akkor a hozzá rendelt slot automatikusan végrehajtásra kerül. A módszer legfőbb előnye, hogy a jelet kibocsátó osztálynak nincs szüksége arra, hogy ismerje a fogadó osztályokat, amely sokkal rugalmasabb és modulárisabb programot tesz lehetővé. Ezeket a funkciókat a Meta-Object rendszer támogatja, amely biztosítja mechanizmus működését, kezeli a futásidejű típusinformációkat, valamint lehetővé teszi a dinamikus tulajdonságkezelést. A rendszer alapját a Meta-Object Compiler (MOC) képezi, amely elemzi a projekt forráskódját, és ha egy osztályban Q_OBJECT makrót talál, automatikusan generál egy MOC fájlt, amely tartalmazza a szükséges kiegészítő kódot a fenti funkciók megvalósításához.

7.2.2 Felépítés

A program két fő komponensből áll: egy megjelenítési rétegből és egy üzleti logikát kezelő rétegből. Előbbi a felhasználói felület megjelenítését végzi, és QML-ben, a Qt deklaratív, JavaScript-alapú nyelvén készült. Ez a réteg felelős a felhasználói interakciók kezeléséért és azok továbbításáért az üzleti logika felé.

Az üzleti logikát kezelő réteg C++ nyelven íródott, és feladata a mikrovezérlővel való kommunikáció, a beérkező adatok feldolgozása, számítások végrehajtása, valamint a fő állapotgép működtetése.

7.2.3 Felhasználói felület

A QML deklaratív jellegéből adódóan a felhasználói felületet hierarchikusan definiált komponensekből építhetjük fel. A QML-ben is jól alkalmazható a signals and slots mechanizmus. Erre jó példa, az általam írt gomb komponens: A komponensben definiálunk egy clicked signalt:

```
QtQuick.Item {
    id: _root

    signal clicked(var event)
.
.
```

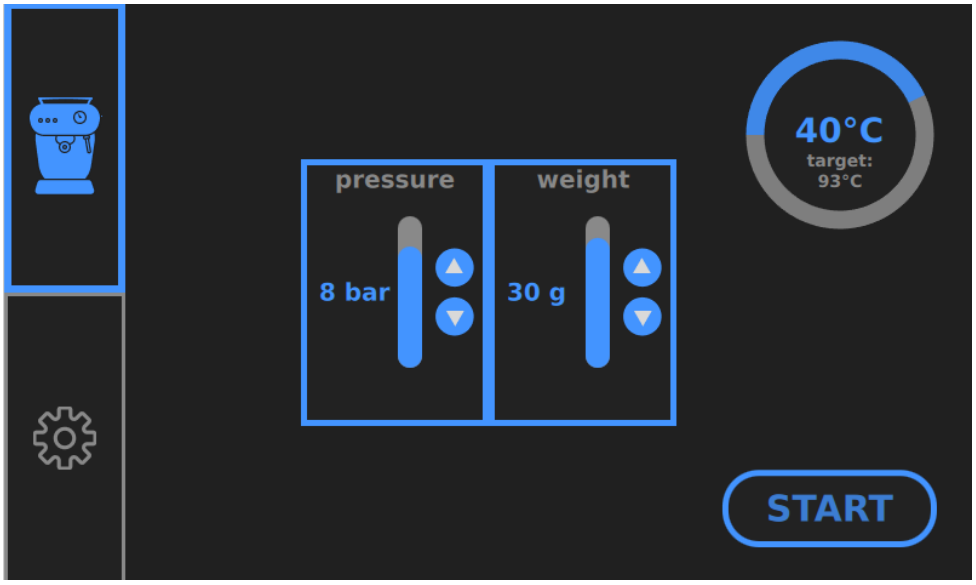
A komponens használatakor, az alábbi példa esetén a főképernyőn egy slotot definiálunk:

```
QtQuick.Item {
    id: _mainItem

    Components.Button {
        onClicked: () => {
            console.log("Button clicked!")
        }
    }
.
.
```

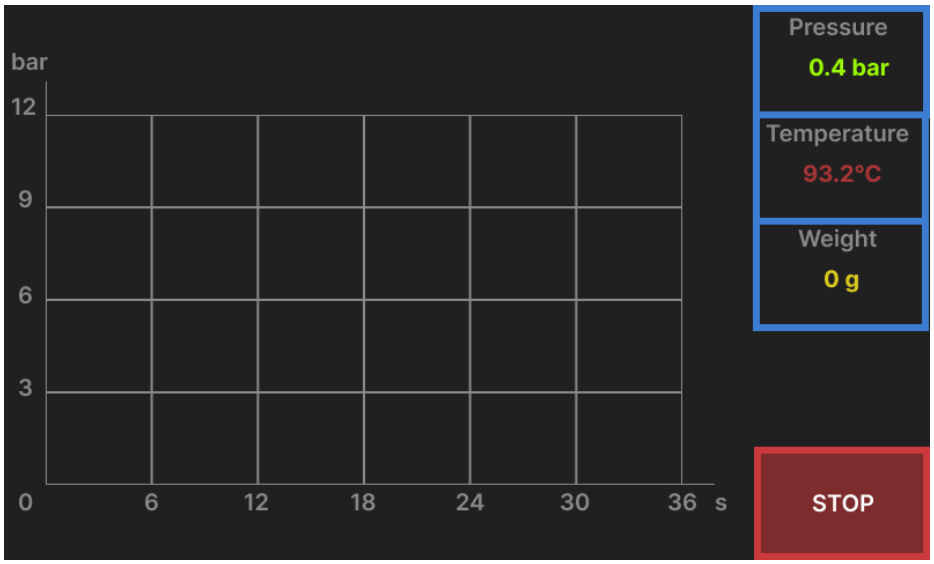
A gomb komponens implementációjában nem kell tudnunk, hogy megnyomásakor keletkező signal-ra milyen slotok fognak reagálni. Így a komponens kódján nem kell változtatnunk, függetlenül attól, hogy hányszor és hol használjuk.

Így végül a kávéfőzőhöz az alábbi felhasználói felületet készítettem:



7.8. ábra Kávéfőző felhasználói felület

A start gomb megnyomása után az állapotot „BREW” módba váltjuk ezzel megindítva a kávéfőzést. Ekkor a felhasználói felület az alábbi képernyőre vált, amelyen láthatjuk a főzési paraméterek kirajzolását. Sajnos ennek a képernyőnek az implementációját nem sikerült befejezni, így a kirajzolás még nincs megvalósítva.



7.9. ábra Kávéfőzés során megjelenő kijelző

7.2.4 Üzleti logika

Az üzleti logikát végző rész fő feladata a feladata a mikrokontrollertől az adatok lekérése és az adatok alapján az állapotgép megfelelő állapotba léptetése. További feladata az felhasználói felület adatainak és eseményeinek kezelése, az események alapján a megfelelő parancsok küldése a mikrokontrollernek.

A rendszer állapotgép diagramját és az állapotok között váltás feltételeit a 7.1.3-as fejezetben bemutattam. A komponens különböző állapotokban, a mikrovezérlőtől eltérő időközönként, más-más üzeneteket kér el, erről a 5.2.2-es fejezetben írtam. A komponens a beállított hőmérséklet, a hibakód és a felhasználói felület eseményei alapján a megfelelő állapotba lép. Az állapotok közötti váltás logikája állapotgép diagramból kiolvasható. Ez a program komponens jelenleg csak az alap funkcionalitásokkal rendelkezik és tovább fejlesztés alatt áll.

Mint a fejezet bevezetésében írtam ez a komponens C++-nyelven van írva, de a felhasználói felület QML nyelven. Ezért a két réteg közötti kommunikációt az alábbi módon oldottam meg.

A kommunikáció a felhasználói felülettel úgy valósul meg, hogy a paramétereket tároló C++ osztályt a Q_OBJECT-ből származtatjuk. A változókat Q_PROPERTY segítségével definiáljuk, és azok módosulásakor signalokat bocsájtunk ki.

```
class SystemStatus: public QObject
{
    Q_OBJECT
    Q_PROPERTY(float currentTemperature READ getTemperature NOTIFY
currentTemperatureChanged)

public:
    float currentTemperature
    void setCurrentTemperature(float temperature) {
        if (m_currentTemperature != temperature) {
            m_currentTemperature = temperature;
            emit currentTemperatureChanged(); }}
signals:
    void currentTemperatureChanged();
};
```

6. kódrészlet C++ osztály adatainak frissítése QML-ben

Ezek után létrehozunk egy singleton objektumot az osztályból, majd regisztráljuk a típust QML-be:

```
std:shared_ptr<SystemStatus> singletonInstance = std:make_shared
<SystemStatus>();

qmlRegisterSingletonType<SystemStatus>("SmartCaso.Providers", 1, 0,
"SystemStatus", systemStatusSingletonProvider);
```

7. kódrészlet C++ osztály regisztrálása QML-be

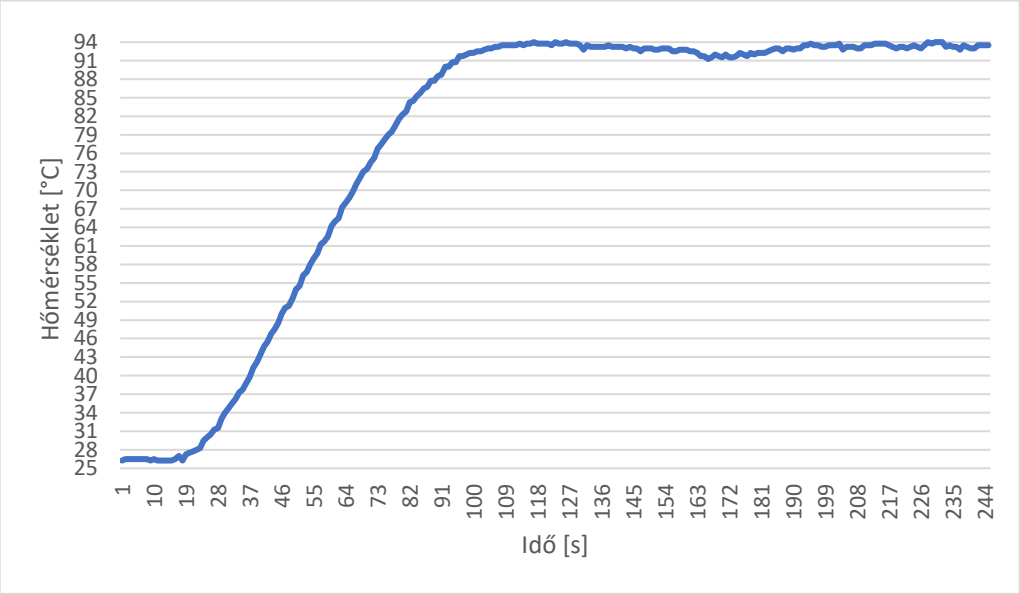
Ezek után QML-ből hozzáférünk a kívánt adatokhoz:

```
text: SystemStatus.currentTemperature
```

A felhasználói felületről érkező események kezelésére lehetőség van, ha a QML-be regisztrált osztály függvényét Q_INVOKABLE makróval látjuk el. Így a függvényt közvetlenül a QML-ből is meghívhatjuk, ami lehetővé teszi, hogy egyszerűen reagáljunk a felhasználói interakciókra, és azok hatására állapotváltozásokat kezeljünk.

8 Eredmények

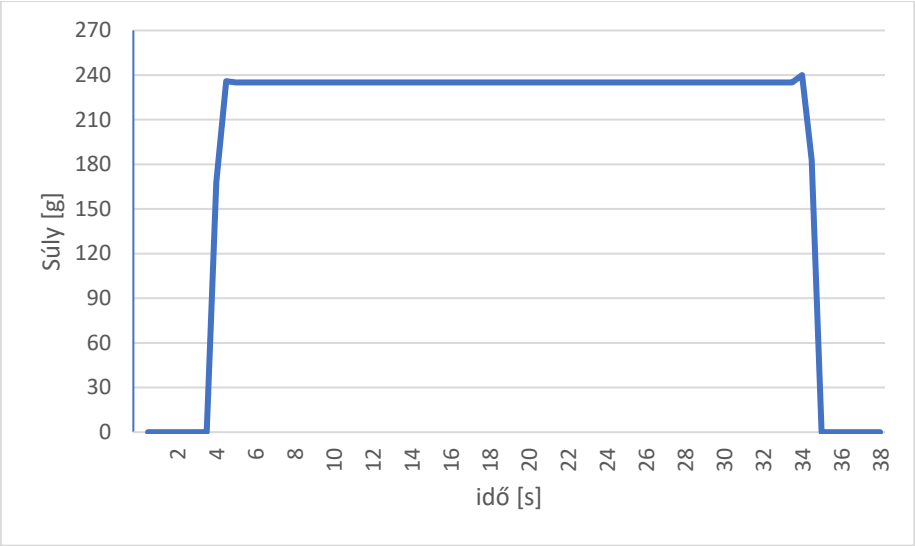
NYÁK forrasztása során az 1 mA-es áramforráshoz szükséges feszültség referencia meghibásodott, és a cserealkatrész nem érkezett meg időben. Ezért a mérést végül egy hőelemmel és egy MAX6675 modullal végeztem. A méréshez a bojlert 93 °C-ra állítottam és 1 másodperces mintavételezési idővel vizsgáltam a hőmérsékletét. A diagramból látható, hogy a hőmérséklet körülbelül ±1,5 °C-ot oszcillál. Ez a PID paraméterek megfelelő megválasztásával javítható lenne, a jövőbeli fejlesztési tervek között szerepel a szabályozó finomhangolása.



8.1. ábra Bojler hőmérséklet felfutási görbe

A mérleget két különböző módon teszteltem, és eredményeit egy 0,1 g pontosságú konyhai mérleggel hasonlítottam össze. Az első teszt során statikus terhelést helyeztem a mérlegre:

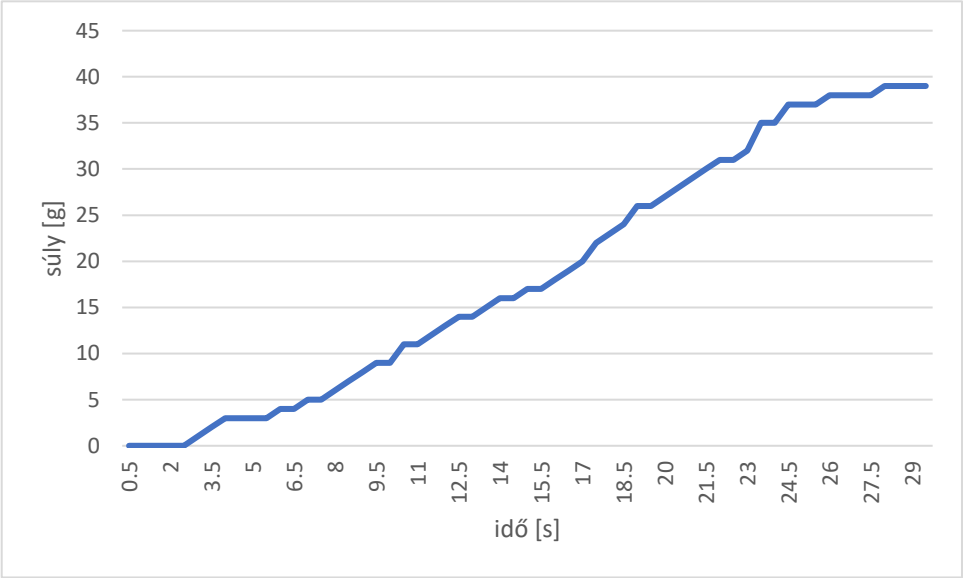
8.2. ábra Mérleg tesztelése statikus terheléssel



8.3. ábra Mérleg mérési adatok statikus terhelés esetén

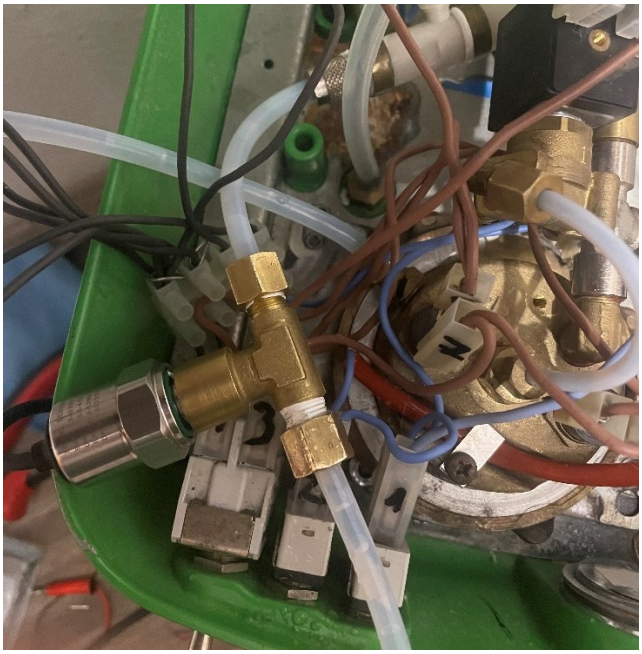
A mérés eredményeként a mérőcellákból készített mérleg 235 g-ot míg a konyhai mérleg 236,8 g-ot mutatott.

A második teszt során a kávéfőzőn vizet engedtem át, miközben a beépített mérleggel mértem a lefolyó folyadék súlyát. A mérés végén megmértem a folyadék súlyát a konyhai mérleggel, amely 39,6 g-ot mutatott, míg a beépített mérleg 39 g-ot.



8.4. ábra Mérleg mérési adatok víz lefolyása során

A nyomásszenzor csatlakoztatásához egy vágógyűrűs T-csatlakozót használtam, de ez nem tartotta megfelelően a nyomást a teflon cső és csatlakozó találkozásánál. Már 1,5 bar víznyomásnál szivárogni kezdett, így a nyomásmérés és nyomásszabályozás tesztelésére nem volt lehetőségem. A szivattyú szabályozása működik, de mérési eredmények hiányában nehéz megállapítani, mennyire effektív a szabályozási metódus. A megfelelő megoldás megtalálása a jövőbeli fejlesztési tervek része.



8.5. ábra Vágógyűrűs T-csatlakozó

9 Összefoglalás

A dolgozat célja egy modern kávéfőző gép egyedi szoftveres és hardveres megoldásainak megtervezése és megvalósítása volt, amely lehetővé teszi a hőmérséklet és a nyomásszabályozását, valamint a lefőzött kávé mennyiségének beállítását A célkitűzések közé tartozott a hőmérséklet ±1 °C-os és a nyomás ±0,5 bar-os szabályozási pontossága, valamint a lefőzött kávé mennyiségének mérése a csepegtető tálcába épített mérlegcellákkal.

A feladat elvégzését a kávéfőző felépítésének és a paraméterek mérési technológiáinak tanulmányozásával kezdtem. A folyamat során rengeteget tanultam a hőmérséklet és nyomás szenzorok működéséről, valamint a gyakorlatban használt mérési technikákról.

A projekt kidolgozását a nyomtatott áramköri lap megtervezésével folytattam, ami a vártnál jóval több időt vett igénybe. A tervezés során komoly kihívást jelentett a kávéfőzőben rendelkezésre álló hely korlátozott mérete. Először egy olyan tervet dolgoztam ki, amelyben a mikrokontroller is egy fejlesztői kártyán csatlakozott az áramköri laphoz. Azonban az így tervezett áramköri lap túl nagynak bizonyult, ezért újratervezésre volt szükség. A folyamat során sokat tanultam a hardvertervezésről, a megfelelő komponensek kiválasztásáról, valamint arról, hogyan kell olyan tervet készíteni, amely a gyártás során is megvalósítható.

A hardvertervezés után párhuzamosan dolgoztam a mikrovezérlő és a Raspberry Pi szoftverének fejlesztésén. Mivel a két eszköz Protocol bufferen kommunikál egymással így a két komponens szorosan összekapcsolódik. A korábbiakban nem foglalkoztam CMake-el ezért a Raspberry Pi projektének fordítása kihívást jelentett. A mikrovezérlő fejlesztése során az STM32Cube grafikus kódgenerálója jelentős segítséget nyújtott. Az egész projekt során a számomra legérdekesebb szakasz a mikrovezérlőn futó különböző folyamatok szinkronizálása és optimalizálása volt FreeRTOS segítségével.

A projekt végső szakaszában az alkatrészek összeszerelésére került sor. Ezt megelőzően mindent fejlesztő kártyákon jumper kábelek segítségével végeztem. Az összeszerelés során azonban több probléma is felmerült: a nyákon a feszültségreferencia túlmelegedett és meghibásodott, a mérlegcellák nem voltak elég stabilak, és a vágógyűrűs

T-csatlakozó nem volt képes a nyomás tartására. Ezek közül az első két problémát ideiglenes megoldásokkal sikerült kezelnem.

A mérések alapján a hőmérséklet-szabályozás közel áll az előzetesen kitűzött célokhoz. A csepegtetőtálcába épített mérleg is megfelelő pontossággal működik, de továbbra is problémát jelent, hogy nagy terhelések esetén a mérleg pereme feltámaszkodik a kávéfőző házára. A nyomásmérést és szabályozást az említett okok miatt végül nem tudtam tesztelni, a rezgő szivattyú szabályozása működőképesnek látszik, de mérési eredmények hiányában nehéz megállapítani, hogy mennyire effektív ez a szabályozási módszer.

Fejlesztési tervek:

- Nyomás szenzor megfelelő csatlakozása: A nyomás szenzornak olyan csatlakoztatást kell biztosítani, amely képes tartani a nyomást. Továbbá a nyomás szenzort érdemes lenne a szivattyútól minél távolabb helyezni, hogy az impulzusok minél kisebb zajt vigyenek a mérésbe.
- Nagyobb mérlegcellák választása: A jelenlegi mérlegcellák nem nyújtanak megfelelő stabilitást, ezért ezeket le kell cserélni nagyobb cellákra. Ez a 3D modellek újratervezését is igényli.
- Hőmérséklet-szabályozás finomhangolása: A hőmérséklet-szabályozás PID paramétereinek finomhangolása, vagy további szabályozási algoritmusok kipróbálása.
- Felhasználói felület fejlesztése: A jelenlegi Raspberry Pi implementáció csak a főképernyőt tartalmazza. A kávékészítési folyamat során megjelenítendő görbék és egyéb funkciókat biztosító képernyők kidolgozása még fejlesztés alatt áll.
- Szoftveres optimalizációk: A mikrovezérlőn futó kódban számos folyamat optimalizálható lenne. A mérlegcellák kiolvasása például blokkoló függvényekkel használatával történik.

Irodalomjegyzék

[1] Ascaso: *ASCASO DREAM ESPRESSO MACHINE,* https://www.youtube.com/watch?v=cvLgvM4WB-E

[2] Home Barsita forum: *Repairing a ULKA vibratory pump,* 2024 október, https://www.home-barista.com/repairs/repairing-ulka-vibratory-pump-t1079.html

[3] Bojtos Attila: *Szenzortechnika előadás jegyzet*, Budapesti Műszaki és Gazdaságtudományi egyetem, 2023

[4] Sapientia Erdélyi Magyar Tudományegyetem: *Jelérzékelők 2. laboratóriumi gyakorlat,* 2024 szeptember, https://www.ms.sapientia.ro/~elektronika/fileok/jelerzekelok/szt_lab02_termistor_v1.pdf

[5] Joseph Wu: A *Basic Guide to RTD Measurements,* 2024 február, https://www.ti.com/lit/an/sbaa275a/sbaa275a.pdf, Texas Instruments

[6] ES Systems: *MEMS Capacitive vs Piezoresistive Pressure Sensors – What are their differences?* Capacitive & Piezoresistive Pressure Sensors - Differences | ES Systems

[7] Avnet Abacus: *Capacitive pressure sensors,* 2024 november, https://my.avnet.com/abacus/solutions/technologies/sensors/pressure-sensors/core-technologies/capacitive

[8] Dr. Halmai Attila, Dr. Samu Krisztián: *Mikromechanika 3. fejezet,* https://mogi.bme.hu/TAMOP/mikromechanika/ch03.html, 2014

[9] XIDIBEI: *XDB401 Industrial Piezoresistive sensor*, 2024 október, https://www.xdbsensor.com/xdb401-industrial-piezoresistive-pressure-sensor-product

[10] ANYLOAD: *How Does a Load Cell Work* https://www.anyload.com/how-does-a-load-cell-work/

[11] Joseph Wu: *A Basic Guide to I2C*, 2024 szeptember, https://www.ti.com/lit/an/sbaa565/sbaa565.pdf, Texas Instruments

[12] FreeRTOS hivatalos dokumentáció, 2024 szeptember, https://www.freertos.org/Documentation

[13] Google: Protocol Buffers Documentation 2024 szeptember, https://protobuf.dev/

[14] R. Mark Stitt: *Make a precision current source*, 2024 február, https://www.ti.com/lit/an/sbva001/sbva001.pdf

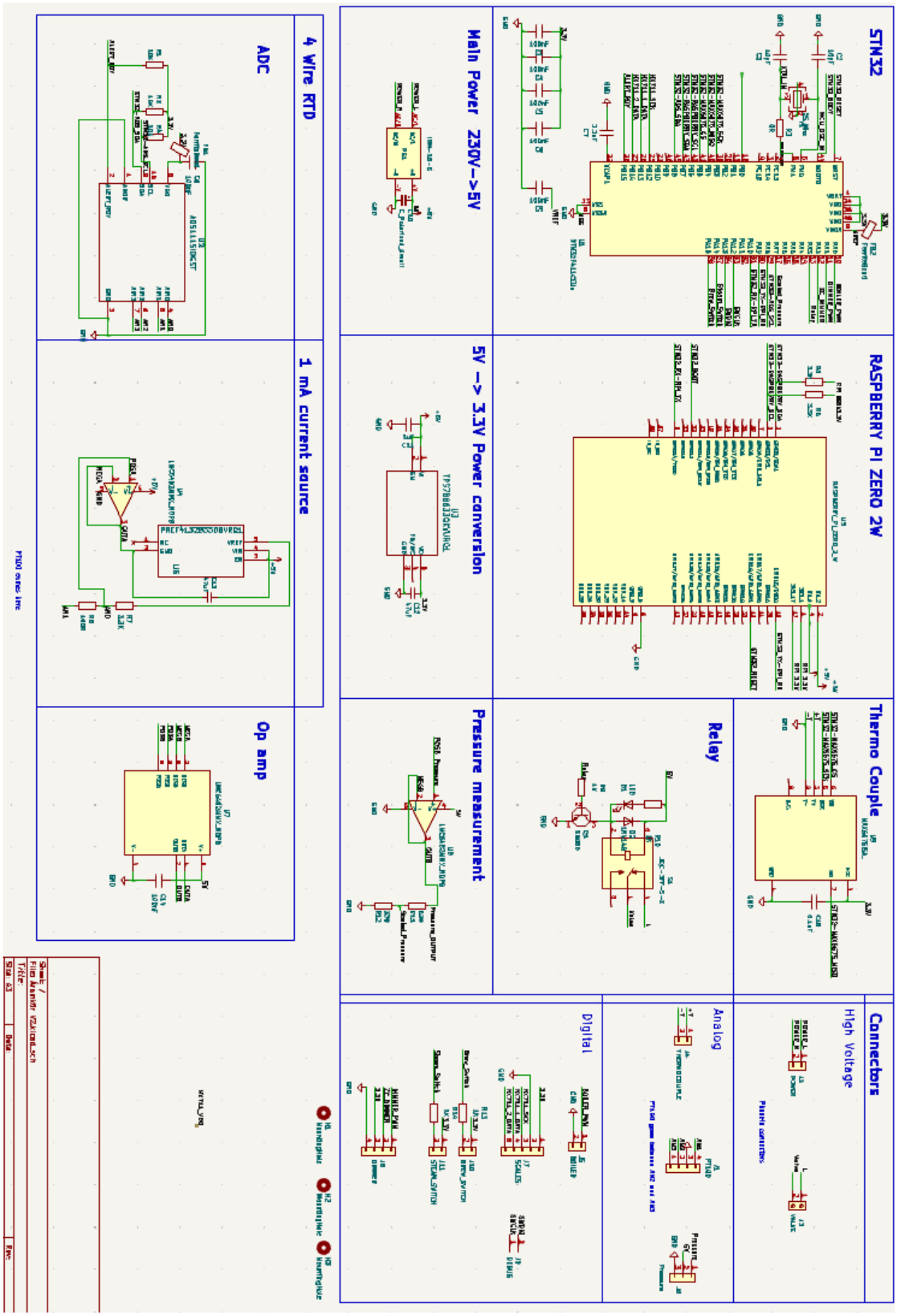
[15] Texas Instruments: *ADS1115 Datasheet,* 2024 október, https://www.ti.com/lit/ds/symlink/ads1115.pdf

[16] Sparkfun*: HX711 Datasheet* 2024 február, https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

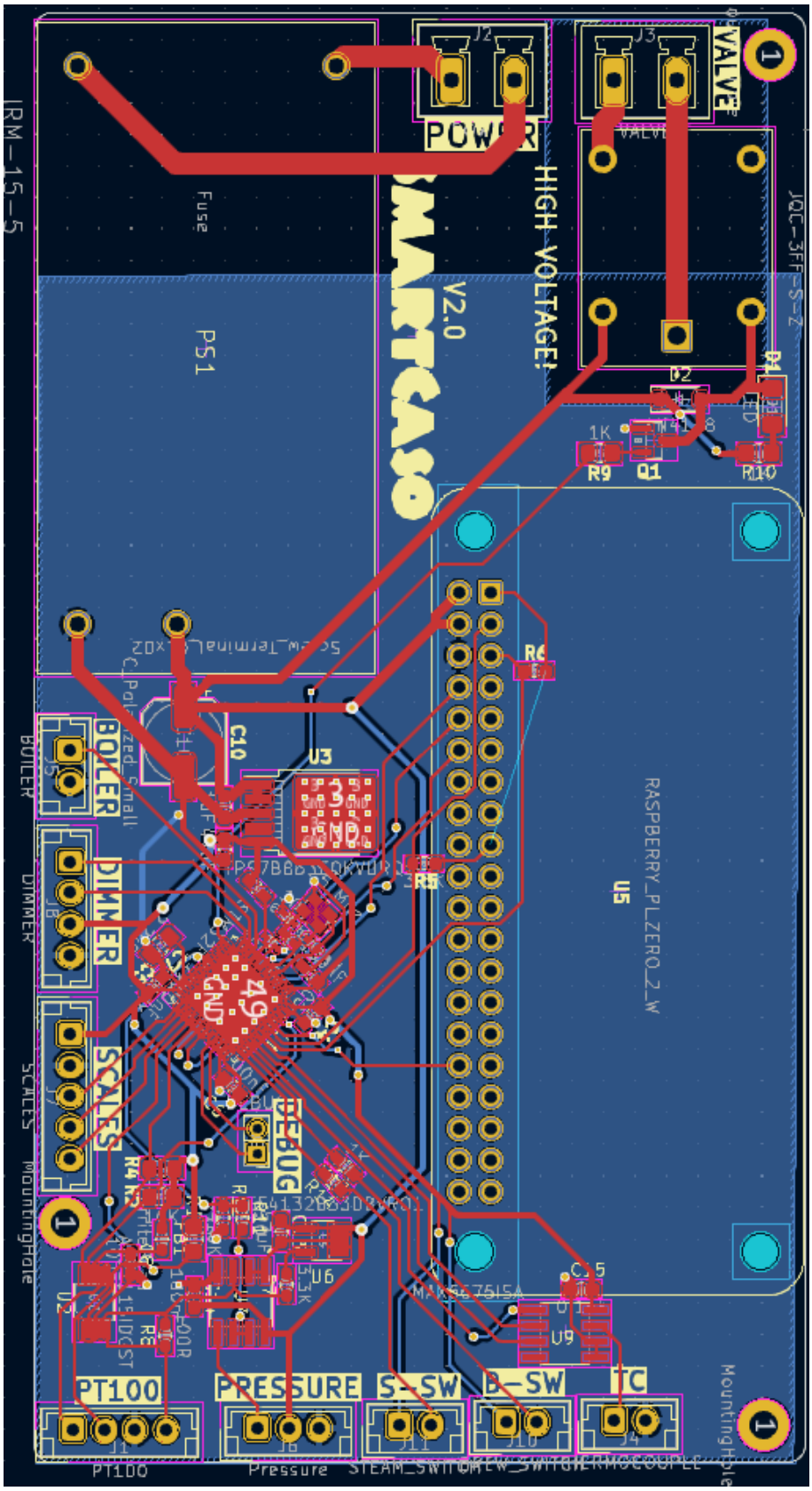
[17] Qt: *Documentation,* 2024, november, https://doc.qt.io/

Függelék

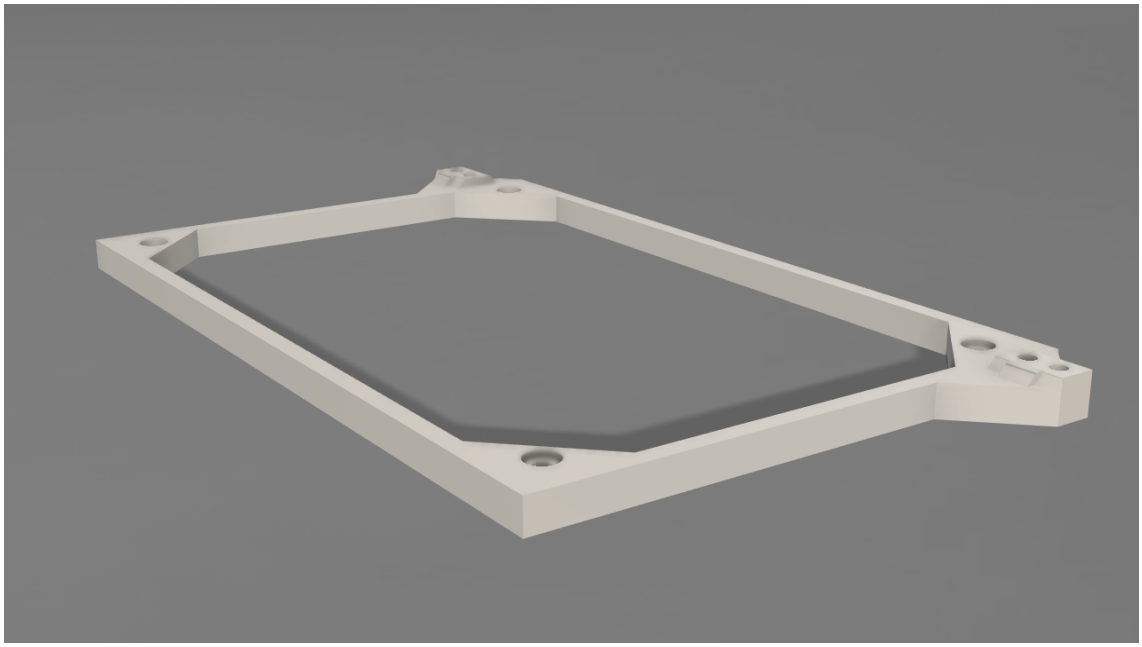
F.1 Kapcsolási rajz



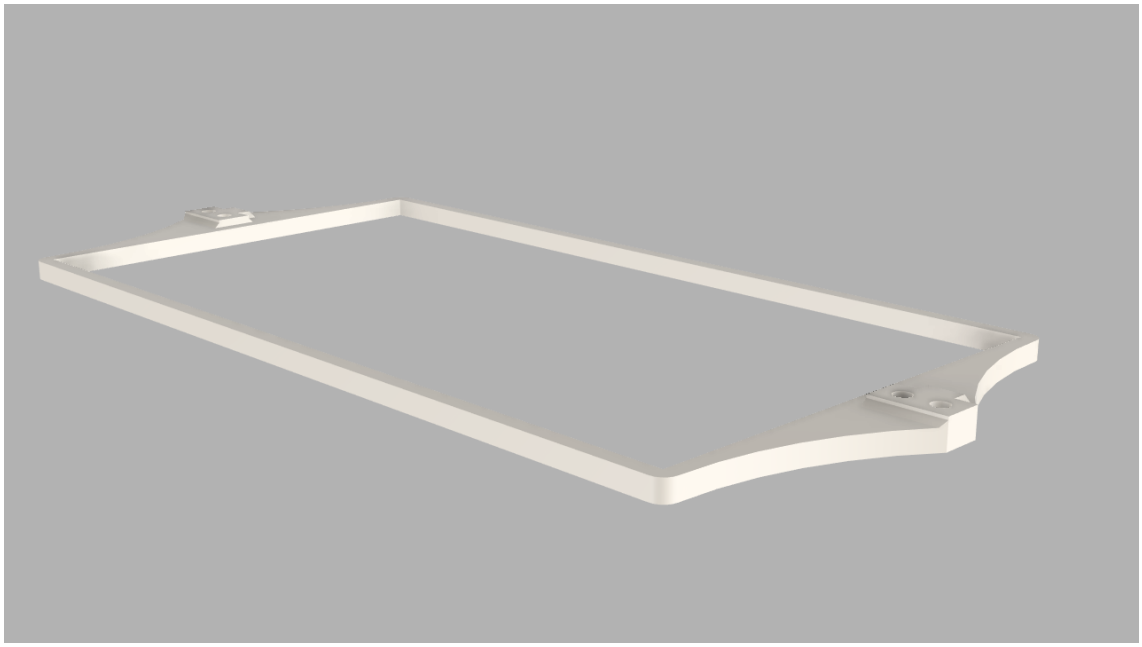
F.2 Huzalozási terv



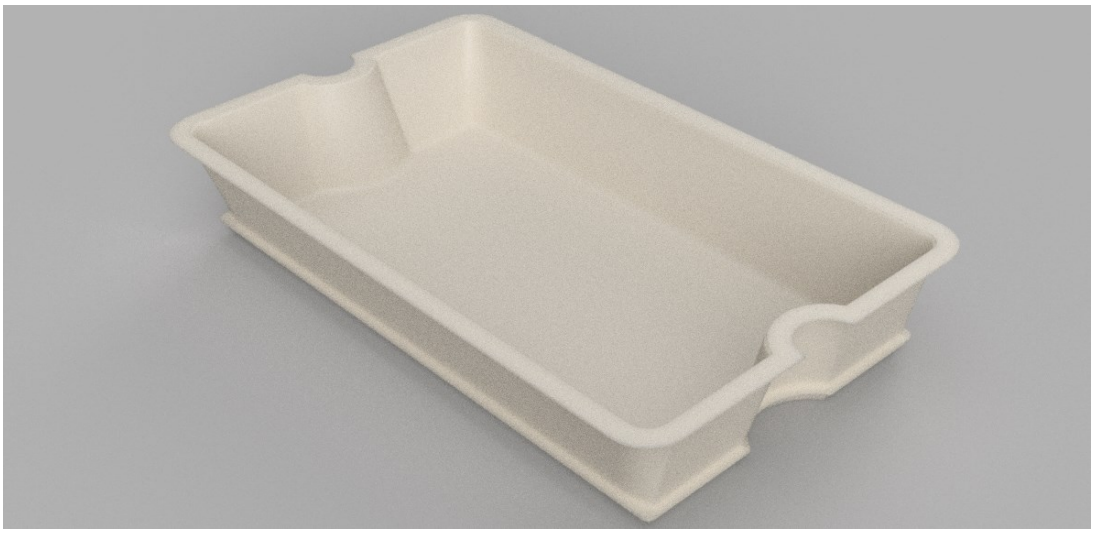
F.3 Mérleg alap 3D modell



F.4 Mérleg csepegtető tálca keret 3D modell



F.5 Csepegtető tálca 3D modell



F.6 Nullátmenetet érzékelő „dimmer”