

UNIVERZITA J. SELYEHO – SELYE JÁNOS EGYETEM
EKONOMICKÁ FAKULTA – GAZDASÁGTUDOMÁNYI KAR
KATEDRA APLIKOVANEJ INFORMATIKY – ALKALMAZOTT INFORMATIKA TANSZÉK

**Vedecká aktivita študentov
Tudományos diákköri munka**

Mikrokontrollérom riadený systém snímania
vybraných fyzikálnych veličín

Mikrokontrollerrel vezérelt rendszer
kiválasztott fizikai mértékegységek mérésére

Vedúci práce:
Témavezető: Ing. Ondrej Takáč, PhD.

Autor:
Szerző: Rácz Gábor

Denné štúdium
Nappali képzés

**KOMÁRNO
2014
KOMÁROM**

Absztrakt

Rácz, Gábor: Mikrokontrollerrel vezérelt rendszer kiválasztott fizikai mértékegységek mérésére Rácz Gábor - Selye János Egyetem, Komárom. Gazdaságtudományi Kar. Alkalmazott informatika Tanszék. Képzési szint: Bc. Témavezető: Ing. Ondrej Takáč, PhD., Komárom 2014. 32 s.

Mikrokontrollerrel vezérelt rendszer kiválasztott fizikai mértékegységek mérésére

A dolgozat a fizikai mértékegységek elméleti és gyakorlati felhasználásának lehetőségeivel foglalkozik, kísérletet tesz a felvetett kérdések válaszainak bizonyítására és a mikrokontroller által irányított rendszerek beüzemeltetésével, ipari alkalmazásával. Továbbá foglalkozik a mikrokontrollerek és a programozásuk megismerésével, programozási környezetük kiválasztásával, belőlük fejleszthető eszközök megvalósításával és egy minta rendszer megépítésével, a hozzá tartozó program beprogramozásával. A dolgozat 5 fejezetből áll, 5 ábrát, 16 táblázatot, 3 mellékletet tartalmaz.

Az első fejezet történelmi áttekintést nyújt, egészen az első elektromos számítógépektől kiindulva, a processzorok kifejlődésén keresztül, a processzorok személyi számítógépekké és mikrovezérlőkké alakulásáig. Közben bemutat néhányat a processzorok jelentősebb típusaiból.

A második rész felállít egy problémakört a dolgozat témája alapján. Specifikálja a kifejlesztendő rendszertől elvárt funkcionálisokat.

A harmadik rész összegyűjti a felvetett problémához szükséges alapokat és eszközöket, kiválasztja a megfelelőket, és ezek alapján megtervez és megépít egy a megfogalmazott problematika megoldásának eleget tevő berendezést.

Az összegzés rész elemzi az elért eredményeket és felvázolja a továbbfejlesztés lehetőségeit.

Kulcsszavak: mikrokontroller, vezérelt, mérés, mértékegység, ipari alkalmazás, páratartalom, hőmérséklet

Abstrakt

Gábor Rác: Mikrokontrolérom riadený systém snímania vybraných fyzikálnych veličín Gábor Rác – Univerzita J. Selyeho, Komárno, Ekonomická fakulta, Katedra aplikovanej informatiky. Stupeň vzdelania: Bc. Vedúci práce: Ing. Ondrej Takáč, PhD., Komárno 2014. 32 s.

Mikrokontrolérom riadený systém snímania vybraných fyzikálnych veličín

Práca sa zaoberá teoretickým a praktickým využívaním fyzikálnych veličín a ich možnosťami. Hľadá odpovede na otázky a dôkazy na riadený systém mikrokontrolérov v priemyselnej praxi. Ďalej sa zaoberá spoznávaním mikrokontrolérov a ich programovaním, vyberaním vývojového rozhrania, možnosťami rozvoja mikrokontrolérom riadených prístrojov a zostavením a naprogramovaním ukážkového prístroja.

Práca je rozdelená do 5 kapitol, obsahuje 5 obrázkov, 16 tabuliek a 3 prílohy.

Prvá kapitola obsahuje prehľad dejín od prvých elektronických počítačov, cez vývin procesorov až do rozdelenia procesorov osobných počítačov a mikrokontrolérov. V rámci toho predstaví niektoré významnejšie typy procesorov.

V druhej časti je stanovená problematika podľa zadania práce, špecifikuje očakávanú funkcionálnu navrhovaného systému.

Tretia časť zosumarizuje potrebné podklady, pomôcky a súčiastky, a z vybraných častí navrhne a zostaví systém, ktorý bude spĺňať riešenie uvedenej problematiky.

Posledná časť kompletizuje a analyzuje dosiahnuté výsledky a naznačí možné smery ďalšieho vývoja.

Kľúčové slová: mikrokontrolér, riadenie, meranie, veličina, priemyselné využitie, vlhkosť, teplota

Abstract

Gábor, Rác: Microcontroller-controlled system for measuring selected physical units Gábor, Rác– Selye János Univerzity, Komárno, Fakulty of Economic, Department of Aplicated Informatics. Leader: Ing. Ondrej Takáč, PhD., Komárno 2014. 32 p.

Microcontroller-controlled system for measuring selected physical units

The thesis deals with the theoretical and practical possibilities of the use of physical units and will attempt to demonstrate and controlled by the microcontroller adjusting systems , industrial use of the responses of the issues raised . It also deals with the microcontrollers and their programming, programming environments for microcontrollers and their implementation plus construction of the system and developing a program for demonstrations. The thesis consists of five chapters, within the chapters we find five figures, sixteen tables and three annexes.

The first chapter contains a historical overview, starting all the way from the first electronic computer , through the development of processors , as they evolved to meet the requirement to be a personal computer or microcontroller. In the meantime the thesis will introduce some of the major processors types .

The second part sets up the role of the microcontrolled system and specifies the required functionalities that need to be implemented .

The third part sums all the instruments together and selects the adequate equipment to design and build a microcontrolled system that satisfies all the problems given in part two.

The summary section gives the summary of the system achievements and outlines the potential for further development .

Key words: microcontroller, controlling, measuring, physical units, industrial use, humidity, temperature

Tartalom

ABSZTRAKT	1
TARTALOM	4
BEVEZETŐ	5
1. TÖRTÉNELMI ÁTTEKINTÉS	6
1.1. A számítógép és processzor kifejlesztése	6
1.2. A számítógéptől a mikrokontrollerig	7
1.3. Mi is az a mikrovezérlő	7
1.4. Mai mikrovezérlők	9
2. PROBLÉMAKÖR MEGFOGALMAZÁSA	11
3. GYAKORLATI MEGVALÓSÍTÁS	13
3.1. Hőmérsékletérzékelő kiválasztása	13
3.2. Relatív páratartalom érzékelő kiválasztása	14
3.3. Processzor kiválasztása	15
3.4. A környezet kiválasztása	20
3.5. Nyomtatott áramkörtábla	21
3.6. Programozási alapismeretek	22
3.7. Program kód	31
3.8. Tesztelés	34
4. ÖSSZEGZÉS	35
BEFEJEZÉS	36
SZAKIRODALOM	37
1. MELLÉKLET: PROGRAM KÓD	40
1.1. melléklet: áramkör bekötési rajza	43
1.2. melléklet: fotódokumentáció	44

Bevezető

Észrevétlenül éltünk minden területére beférkőztek a mikroszámítógépek. Néha nem is tudatosítjuk, hogyan működik a mikro sütő, mi irányítja a játékautót, mitől jár a digitális óra, vagy éppen mi dolgozik a mobiltelefonunk, mp3 lejátszónk és a tévénkben. Belénk nevelte a fogyasztói társadalom, hogy szeressük az intelligens termékeket. Azonban mi működteti, hogyan lesz intelligens? Teheti fel a kérdést az, akit érdekelnek a miérték és a hogyanok. A válasz egyszerű: a mikroszámítógéptől, amit beletettek, meg valahogy beleprogramozták azt az intelligenciát. Lehet, a többségnek elegendő ez a válasz, de vannak, akik mélyebben szeretnék ismerni működésüket. Azért választottam munkám témájaként a mikroprocesszorok programozását, mert nem elégedtem meg a válasszal: hogy mert működik... Minél többet tudtam meg a miértokról, annál több további kérdés merült fel bennem.

Munkám célja egy mikroprocesszor elméleti megismerése és gyakorlati felhasználásának kivitelezése.

1. Történelmi áttekintés

1.1.A számítógép és processzor kifejlesztése

1946-ban egy pennsylvániai egyetemen Neumann János tervei alapján megépítik az első elektronikus számítógépet az ENIAC-ot. Ez az alapja a ma használatos számítógépeknek, melyeket a Neumann-elvű számítógépeknek is nevezünk. Az ENIAC (Electronic Numerical Integrator And Computer) 18 ezer elektroncsövet, 70 ezer ellenállást, 10 ezer kondenzátort, 1500 jelfogót és 6 ezer kapcsolót tartalmazott. Méretei 33m hosszú 3,3m magas, 1 méter széles, 30 tonnát nyomott, 220m² helyet foglalt el és 800kW-ot fogyasztott. 5000 összeadást vagy néhány száz szorzást végzett el 1 másodperc alatt. Elődje az EDVAC relés számítógéptől 1000-szer volt gyorsabb. [7]

A tranzisztor feltalálása a robbanásszerű fejlődést hozta magával. Kisebb méretével, fogyasztásával, megbízhatóságával és a nyomtatott áramköri lapok megjelenésével a második generációs számítógépek alapjául szolgált. Így nagymértékben megnőtt a számítógépek megbízhatósága és sebessége.

Szakemberek szerint a nyomtatás feltalálásához hasonló jelentőségű előrelépést eredményezett az integrált áramkörök 1962-ben való megjelenése, melyekben egy hordozó szilícium lapkán több tranzisztor és az őket összekötő vezetópályákat is létrehozzák, majd ezeket sorozatgyártással tetszés szerinti darabszámban gyártották.

Ez a harmadik generációs számítógépek megjelenését eredményezte. A tranzisztorokat kiszorítják az alacsony és közepes integráltságú áramkörök, funkcionális egységek alakulnak ki és köztük megjelenik az egységes csatornarendszer.

1971-ben megjelenik az első mikroprocesszor az INTEL 4004. 4 bit adatokkal és 8 bites utasításokkal külön 1kB adat és 4kB program memóriával, 46 utasítást ismert és 2300 tranzisztort tartalmazott 16 lábú DIP tokozásban. Még ebben az évben megjelenik a 8 bites 8008-as.

1974-ben jelenik meg az Intel 8080-as és a 8085 mely már a ma is használt PC-k processzorainak elődjének tekinthető.

1975-ben a Motorola bemutatja a 6800-as processzorát, melyek továbbfejlesztései a 650x jelennek meg a Commodore 64 (6510), Apple (6502) és Atari 2600 (6507) személyi számítógépekben. Ezek a számítógépek már eljutottak hazánkba is, így alkalmunk nyílt a használatára.

1976-ban Zilog kiadja a népszerű Z80-at ami szintén 8080 továbbfejlesztés volt. 2,5MHz, később 8MHz és a CMOS verzió 10MHz sebességgel. Az AMD az Am2901-es processzorával jelenik meg, később 1979-ben az első lebegőpontos processzorral az AMD9611-el, mely képes volt összeadás, kivonás, szorzás, osztás, szinusz, koszinusz számítására, sebessége minden akkori processzornál gyorsabb volt. [22]

1.2.A számítógéptől a mikrokontrollerig

1974-ben megjelenik az első mikrovezérlő a Texas Instruments 4 bites TMS 1000-es, melyen egy tokban adat RAM, és program memóriát ROM valamint I/O egységeket tartalmazott, így lehetővé téve az önálló működést.

1977-ben jelenik meg az Intel 8048-as mikrovezérlő. Az adatmemória a token belül volt, de a programot külső ROM-ban kellett tárolni. Ezt volt hivatott orvosolni a 8051, és a 8052. Az adatmemória 128 bájtos volt mely közösen címezhető a program memóriával. Népszerűségét olcsóságának is köszönheti, több mint 1 Billiót adtak el belőle.

1.3.Mi is az a mikrovezérlő

A mikrovezérlő tehát egy erőforrásokban korlátozott önmagában működőképes számítógép. Míg a modern számítógépekben a közös gyökerektől indulva hatalmas, robbanásszerűen fejlődik a processzor sebesség, memória méret, óriás háttértár, csatlakoztatható perifériák korlátlan tárháza, addig a mikrovezérlőkben nincs magas órajel, nincs operációs rendszer, nincs méretes háttértár és a perifériákat is a programozónak kell szoftveresen kezelnie. Ennek ellenére széles körben használunk mikrokontrollereket, mivel egyes megoldások az egyszerűség, vagy csupán a kis méret végett nem igényelnek bonyolult számítógépeket, ezek pl. a különféle riasztók, távirányítók, de háztartásunk majdnem minden berendezése, mely már elektronikus vagy digitális vezérlést tartalmaz: mikro sütő, mosógép, mosogatógép.

A mikrokontrollereket általában valós idejű feladatokra használják, ahol valamilyen külső inputra nagyon rövid időn belül (milliszekundumokon belül) reagálni kell. Beépített perifériái lehetnek:

1. oszcillátor: órajelet ad a processzor számára, ezen kívül egy számláló vezérlésével időmérésre is használható
2. ROM a vezérlőprogram tárolására és RAM futás közbeni adatok tárolására
3. számlálók/időzítők: ezeket külső és belső jelek segítségével lehet léptetni. Használhatók egyszerű impulzusszámlálási feladatokra és frekvenciamérésre is.

4. watchdog időzítő: biztonsági eszköz a működtető program "lefagyása" ellen. Egy időzítő, melyet a szoftvernek periodikusan nullázni kell. Ha nem teszi, akkor a watchdog áramkör hardveresen újraindítja a mikrokontrollert, amitől talán elmúlik a lefagyást előidéző állapot.
5. EEPROM a kalibrációs adatok vagy más paraméterek, és adatok tárolására. A tápfeszültség elvesztése után is megőrzi a tartalmát.
6. DSP (digitális jel-processzor): a nagysebességű digitális jelfeldolgozáshoz szükséges különleges számítási műveleteket gyorsító számolóegység
7. Jelátalakítók:
 - a. Analóg-digitális átalakító (A/D konverter) a külső analóg jeleket (feszültséget) digitális formába, a processzor által feldolgozhatóvá alakítja
 - b. Digitális-analóg átalakító (D/A konverter) a belső digitális jeleket alakítja át analóg jelekké.
 - c. Komparátor: két analóg feszültséget hasonlít össze.
8. Kommunikációs interfészek buszok: a különböző eszközök egymással történő kommunikációját valósítják meg.
 - a. I2C (Inter-Integrated Circuit) busz,
 - b. UART (Universal Asynchronous Receiver/Transmitter),
 - c. SPI (Serial Peripheral Interface) busz,
 - d. CAN (Controller Area Network) busz,
 - e. 1-Wire,
 - f. USB,
 - g. parallel port külső párhuzamos busz
9. Memória jellegű tárok kezelés
 - a. párhuzamos adat- és címbuszú memóriák kezelése
 - b. SD-kártya kezelése
10. Meghajtó egységek:
 - a. LCD-panel meghajtók
 - b. LED kijelző meghajtók
11. Jelgenerátorok:
 - a. PWM: négyszögjel előállítás szabályozáshoz
12. debug interface: [21]

1.4. Mai mikrovezérlők

A mikrokontrollerek sem álltak meg a fejlődésben. Egyre szélesebb körű támogatásban részesülnek, csatoló felületeik bővülnek, hogy a fejlesztés során minél kevesebb alkatrészszel lehessen megoldani a feladatok legszélesebb skáláját amellet, hogy az eszköz fogyasztását, méretét és költségét minimalizálják.

Napjainkban nagyon sok gyártó fejleszt mikrovezérlőket, ilyen típusok pl.:

1. Microchip PIC (PIC18, PIC24, PIC32)
2. Atmel AVR, ATtiny és ATmega32, Xmega
3. Motorola/Freescale (pl. 68HC, S08, S12, DragonBall, ColdFire, M-CORE, MPC5xx, i.MX családok)
4. Intel MCS-41/42, MCS-48, MCS-51, MCS-96, MCS-296
5. Parallax SX (P8X32A)
6. ARM processzorok, különböző gyártóktól

Sokasodnak a különféle oktatási célból tervezett fejlesztő környezetek, ilyenek pl.:

7. AVR Butterfly (Atmel platform)
8. Arduino környezet (Atmel platform)
9. RaspberryPi (ARM platform, Raspbian op.rendszer)
10. BeagleBone (ARM platform)
11. P8X32A Propeller Demo Board (Parallax platform)

A Microchip Technology által gyártott PIC mikrovezérlők (peripheral interface controller) Harvard-architektúrájú 8 bites RISC típusú mikrovezérlők a legelterjedtebbek közé tartoznak, mint pl. PIC10x, PIC12x, PIC14x, PIC16x, PIC18x, PIC24x, PIC32x. Széles irodalommal rendelkeznek, oktató tananyagoktól kezdve egészen a professzionális szintű ipari felhasználásokig.

Széles körben elterjedtek az AVR az Atmel cég által 1996-ban kifejlesztett módosított Harvard-architektúrájú 8 bites RISC típusú egy chipes mikrovezérlői. A módosított Harvard-architektúra, melyben közös címbuszon osztozik a program és az operatív tár, nagymértékben hasonlít a Neumann-elvű számítógépekhez. Az AVR volt az első mikrovezérlő család, amelyben chipre integrált flash memóriát kezdtek használni a programtárolásra, ellentétben más mikrovezérlőkkel, amelyeknél ekkor még egyszer programozható ROM, EPROM vagy EEPROM memóriák szolgáltak erre a célra.[14]

Nagy népszerűségnek örvendenek az oktatási célú vagy fejlesztői környezetek, vagy hobbi fejlesztőcsomagok is. Az Arduino az ATMEL cég platformjára építve rengeteg előszerelt a gyufaskatulya méretű Arduino Micro, Nano, Mini pro-tól kezdve egészen az Arduino Mega, Pro, Robot-ig, különféle csomagokban szerelve érdekes választás a kísérletezni és tanulni vágyók számára. Saját Arduino-s fejlesztői környezetet biztosítva az eszközeik számára. Hasonló módon a többi gyártó mint a PIC és Parallax is rendelkezik ilyen oktató fejlesztő csomagokkal, hobbi kit-ekkel.

Hasonlóan, a szintén oktatási célból fejlesztett, nevében is játékos Raspberry Pi platform, amely a robbanásszerűen induló és azóta sem lankadó érdeklődést mutató, nagy szurkoló tábornak örvend.

A Raspberry Pi egy bankkártya méretű, egyetlen lapra integrált számítógép, amelyet az Egyesült Királyságban fejlesztettek oktatási célokra. A gép különböző Linux-disztribúciókkal működtethető, illetve elérhető egy RISC OS verzió is. Jelenleg két változata létezik. Az A és B változat. Az A változat nem rendelkezik Ethernet csatolóval, csak egy USB portja van és a rendszer memóriája 256Mb. A B modell 2db USB porttal rendelkezik, 512MB memóriával és integrált Ethernet csatolóval. A hivatalosan ajánlott operációs rendszer a laphoz a Raspbian, ami a Debian Linux kifejezetten RaspberryPi-re optimalizált változata.[25]

Az ARM processzor köré épített mikrokontrollert különféle saját operációs rendszert lehet ellátni, mint pl: Raspbian, Pidora, Openelec, RaspBMC, Archlinux, Noobs, NoobsLite, RiscOS. Digitális HDMI és analóg AV kimenetén bármilyen tv vagy monitorra csatlakoztatható. Kimeneti GPIO portja viszonylag kevés lehetőséget biztosít, ezért szükséges egyéb kiegészítők beszerzése komolyabb feladatokhoz. Ellenben kiegészítő széles választékát lehet az internetről megrendelni, érintőpaneles lcd kijelző, HD web kamera, GSM modul, különböző kimeneti port bővítők és vezérlő panelek. Ezek természetesen nem olcsó termékek, árukat általában tükrözi népszerűségük is.

A BeagleBone projekt a nagyban hasonlít a Raspberry Pi projekthez, saját operációs rendszerrel rendelkező ARM architektúrájú rendszere és széles BE/KI meneti portja viszont már majdnem kielégíti az IPC ipari szabványok által elvárható követelményeket. Kisebb ipari projektek kiváló alapja lehet, ára ezt tükrözi is.

2. Problémakör megfogalmazása

Bármely ipari, gazdasági, polgári létesítmény épületgépészeti problémája a szellőztetés és a fűtés optimális arányának megvalósítása, amellyel biztosíthatjuk megfelelő levegőminőséget és kellő hőmérsékletet.

- elsősorban biztonságosan, hogy a levegő minősége mindig megfeleljen az egészségügyi előírásoknak, és amely alatt azt is értjük, hogy az épületben megfelelő helyen (nem süt rá a nap, nincs fűtőtest vagy tűzhely közelében) felszerelt berendezés legyen, amely által érzékelt a túl magas hőmérséklet, ami 50°C felett tűz jelenlétére utal.

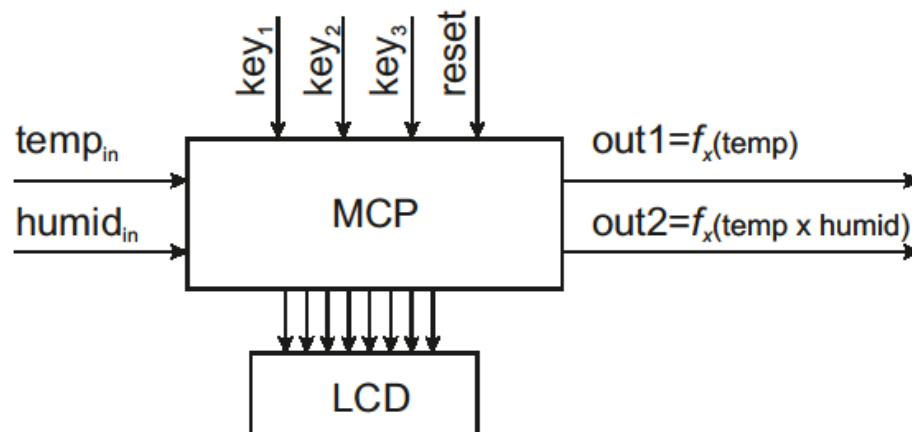
- nem utolsó sorban energiatakarékosan, így megőrizve a feleslegesen kiszellőztetett felmelegített levegőt, ami a mai magas energiaárak és haszonelvű társadalmunk pénzspórolás elvei mellett, természetbarát cselekedet is.

Az állattenyésztésben és a növénytermesztésben szintén elengedhetetlen a megfelelő hőmérséklet fenntartása és a levegőminőség biztosítása. Továbbá a megszerzett adatok folyamatos megfigyelése és elraktározása, s ezek kiértékelve később összevethetők legyenek egyéb hozam eredményekkel, így biztosítva az optimális termelékenység, és ebből következő nyereség mértékét. Ezért fontos az üvegházak klímájának vezérlése, az állattartás épületeinek szellőztetése és fűtése, élelmiszerraktárak és hűtőházak optimális hőmérsékleten tartása. Hasonlóan ipari technológiák is igényelhetnek megfelelően beállított hőmérsékletet és páratartalmat.

Továbbá a civil életben is mindenképpen elengedhetetlen az irodaépületek, bevásárlóközpontok, mozik, előadótermek, sportlétesítmények stb. megfelelő levegőellátása, és ezek intelligens vezérlése, ami alapkövetelmény.

Valószínűleg mindenki találkozott már a tűző napon leparkolt autó felhevülésének problémájával, amelynek szellőztetése enyhíthetné a felforrósodás mértékét, így kellemesebbé téve a mielőbbi beszállást és gyors indulást. Ehhez hasonló probléma lehet a felhevülő számítógépházak megfelelő szellőztetésének biztosítása is.

A cél egy olyan rendszer megépítése, amely egy adott épület vagy zárt tér meghatározott pontjain lévő mérések alapján, a megszerzett hőmérsékletet és a páratartalom adatait kiértékelve, vezérlést biztosít az adott berendezések irányításához. Ilyen lehet pl. a fűtés be-, kikapcsolás, reguláció, automata ablaknyitás, szellőztető rendszer irányítás, riasztás.



1. ábra: Input/output blokk vázlat

Esetünkben a javasolt rendszerünk egy mini-üvegház, de akár egy kisebb méretű üvegház, télikert hőmérsékletének szabályozását fogja megoldani. Ha kihűl vagy túlhevül, vagy ha nagyon kiszárad vagy túlzottan bepárasodik az üvegház, kár eshet a növényekben. Ennek megakadályozására szellőztetést és a fűtés irányítását várjuk el rendszerünktől. Ez a problémára kezelhető egy általunk tervezett kompakt mikrokontroller vezérelt rendszerrel, amely olcsó, biztonságos megbízható megoldás.

A rendszerünk be- és kimeneti eseményeinek vázlatos rajzát az 1. ábra mutatja. Működése a következő lépésekben írható le:

Ha a hőmérséklet értéke egy Tlow beállított érték alá esik, akkor kapcsoljon be a fűtés, és mindaddig maradjon bekapcsolva, míg a kívánt hőmérséklet értéke nem haladja meg a beállított Thigh értéket. Erre azért is szükség van, hogy megakadályozzuk az irányítás túl gyakori ki-be kapcsolódását. A szellőztetés bekapcsolását, hasonlóan a hőmérséklet értékének egy előre beállított Vhigh érték túllépése esetén, vagy a páratartalom értékének a Hhigh túllépése esetén várjuk el, és hasonlóan kikapcsolását a hőmérséklet Vlow érték és a páratartalom értékének Hlow érték alá esése után várjuk el.

Előre beállított értékeink Tlow, Thigh, Vlow, Vhigh, Hlow, Hhigh legyenek egy menün keresztül változtathatóak, beállítás után ezek az értékek legyenek elmentve, és ezeket használja a rendszer, esetleges áramszünet után ezekkel az értékekkel induljon újra. A menü kezelése legyen egyszerű, lehessen az értékeket növelni, csökkenteni, oly módon, hogy a low érték ne lehessen több a high értéknél. Ezt egy háromgombos sorba léptetős menü oldja meg. Ehhez szükségünk van 3 key inputra.

A mért értékek azonnali megjelenítéséhez, a menü eléréséhez, az interakció megvalósításához rendelkezzen valamilyen kijelzővel, esetünkben LCD kijelzővel.

3. Gyakorlati megvalósítás

A dolgozatot hosszas előkészítés, kutató munka és kísérletezés előzte meg. A szakirodalom hosszas tanulmányozása, tapasztalt kollégák véleményeinek meghallgatása és az interneten található hasonló kísérletek adatai és publikációk szolgáltak információforrásként.

A megvalósításhoz elengedhetetlen alapvető elektronikai tudás, az alkatrészek működésének ismerete, és természetesen az elméleten túl a gyakorlatias hozzáállás nélkülözhetetlen az esz közök beszerzéséhez, a rendszer megépítéséhez.

A megvalósításhoz szükségünk lesz mindenképp a következőkre:

1. Forrasztó, forrasztóórn és gyanta
2. Próbapanel, vezetékek, tápegység
3. Alkatrészek, és dokumentációjuk, lásd lejjebb.
4. ISP programozóra <http://www.fischl.de/usbasp/>
5. Programozó szoftverre <http://khazama.com/project/programmer/>
6. PC és AVR Studio <http://www.atmel.com/Images/AStudio61sp2net.exe>

3.1. Hőmérsékletérzékelő kiválasztása

Hőmérséklet érzékelésére több lehetőség van, pl. hőérzékeny ellenállás, termisztor, hőérzékeny félvezető, analóg kimenetű áramkör vagy digitális áramkör.

Termisztor

Maga a termisztor a hőmérséklet függvényében ellenállását változtató ellenállás. Általában nem lineáris függvénnyel írja le a hőmérsékletváltozást, ezért megvalósítása bonyolult: kalibrálás. stabilizált feszültség biztosítása, kiértékelés táblázatból. Felhasználása bonyolultsága miatt lett elvetve.

Analóg lineáris kimenetű hőmérő IC

Ezek a hőmérő szenzorok tipikusan A/D átalakítós megoldásokhoz vannak tervezve. 10mV-os fokenkénti skálázódással és széles tápfeszültség toleranciával rendelkeznek. Népszerűségük, relatív alacsony árak és könnyű beszerezhetőségük miatt kerültek figyelmünkbe.

Digitális kimenetű hőmérő IC

Ezek a hőérzékelő chipek, hasonlóan az analóg kimenetű hőmérőkhöz már előre beállított, kalibrált értékeket adnak vissza, többnyire soros kommunikációval küldve át a megszerzett és digitalizált adatokat, így még kevésbé lesznek érzékenyek az adatok a továbbítás közben felmerülő zajra. Esetünkben ez a megoldás elvetésre került a bonyolultabb megvalósítása miatt.

1. Táblázat: Gyakran használt analóg kimenetű hőmérők legfontosabb adatai

		
Típus	LM35 főbb jellemzők	MCP9700 főbb jellemzők
Gyártó	Texas Instruments	Microchip
Lineáris skálázódás	10mV/°C	10mV/°C
Garantált pontosság	0.5°C (25°C-nál)	2°C (0-70°C)
Kimeneti feszültség	0°C – 0mV	0°C – 500mV
Mérési tartomány	-55°C – 150°C	-40°C – 125°C
Tápfeszültség	4 – 30 V	2,3 – 5,5 V

Forrás: a gyártó által kibocsátott adatlap Texas Instruments [33], Microchip [32]

A két széles körben elterjedt érzékelő közül végül a Microchip gyártmányára esett a választás, mivel ezzel egyéb alkatrészek beszerelése nélkül is mérhetőek a 0°C alatti hőmérsékletek.

3.2. Relatív páratartalom érzékelő kiválasztása

A relatív páratartalom érzékelők kínálata már nem annyira széles, mint a hőmérsékletérzékelőké. A választott hőmérsékletérzékelőhöz hasonlóan 2,7 - 5,5V tápfeszültség tartományban üzemelő alkatrészt kell keresni, az esetleges 3V-os üzemmód biztosításához (minden alkatrésznek működnie kellene 3V-on). A közülük való válogatást főleg az alkatrész ára határozta meg. Látókörükbe került néhány típus, mint a Sensirion SHT1x család, és az ehhez nagyon hasonló a DHT11, melyek 1wire kommunikációval ellátott digitális kimenetű integrált áramkörök, melyek árukat tekintve nem nagyon drágák. Viszont a nem teljes I2C kompatibilis kommunikációjuk és ez általi nehezebb megvalósítás, esetleges felmerülő problémák, további keresgélésre sarkalltak. Így a

választás egy még olcsóbb termékre, a Honeywell HIH 5030-001. analóg lineáris kiértékelésű relatív páratartalom érzékelőre esett. A választást az is befolyásolta, hogy a hőmérsékletérzékelőnk is hasonló analóg lineáris kimenetű integrált áramkör.

3.3. Processzor kiválasztása

Keresgetésünk a két szélesebb körben elterjedt mikrokontroller gyártók Microchip cég PIC termékei és az ATMEL cég mikrokontrolleri körül kezdődött. Ezekhez rengeteg leírás, tananyag, hasonló kidolgozott munkák, szakirodalom található, amelyek tanulmányozása után került sor a döntésre. Így a könnyebb elérhetősége és kisebb ára miatt az Atmel cég gyártmányai körül folytattuk a keresést. Elsődleges szempont, hogy rendelkezzen A/D átalakítóval, mivel ilyen hőmérőre esett a választás. Ezzel az opcióval a mikrokontrollerek majd mindegyike rendelkezik, ez nem szűkíti le a választékot. Továbbá rendelkeznie kell legalább 3x8 input/output lábbal, 3 a nyomógombok, 1 reset bemeneteihez, 8 az LCD panel vezérléséhez, 2 az RS232 kommunikációhoz, 2-3 az ADC átalakítókhoz, 2-3 a kimenetek vezérléséhez, 1 a külső megszakítás-kezeléshez. Szükségeltetnek még belső időzítők, a kiértékelés és az adatok valós idejű feldolgozása szempontjából, szorzó és osztó logika az átlagolás számítására, gyors adatfeldolgozás, azaz frekvencia legyen min. 8MHz, a kalibrációs és elő beállított értékek mentésére flash memória, min. 1kB.

Az Atmel termékei közül weboldalukon válogathatunk egy Microcontrollers Selector adatbázis segítségével. Ez egy haladó mikrokontroller programozó számára sem könnyű feladat. Hosszas kutatás után a lentebb felsorolt néhány családtípus került látókörünkbe:

ATTiny

Ez a kisméretű chip főleg akkor kerül előtérbe, ha az alacsony ár és a kis méret a fontos. Ha a fejlesztés befejeződik és kialakulnak a végleges követelmények, választható egy megfelelő kis párja egy nagyobb ATMega testvérnek helyettesítésére:

ATMega

8-bites mikrokontroller család már bővebb kínálatát nyújtja a be és kimeneti portoknak, nem leszünk korlátozva a tokozás lábainak számával, ami így több periféria beépítését, szélesebb körű felhasználást tesz lehetővé és megkönnyíti a fejlesztést.

XMega

Ez a család a tovább fejlesztett változata az előbbieknek, magasabb teljesítményt nyújtva, kisebb áramfogyasztás mellett. Több új szolgáltatást kínálva, mint pl. a DMA.

2. Táblázat: Processzor családok gyors összehasonlítása

	ATtiny	ATmega	Xmega
Programmemória	0.5-16 kB	4-256 kB	16-384kB
Adatmemória	32-512 byte	0.5-16 kB	2-32 kB
EEPROM	0-512 byte	0,25–4 kB	1-4 kB
tokokban	6–32 láb	24–100 láb	44–64–100 láb
Maximális órajel	4–20 MHz	16–20 MHz	32 MHz
Periféria készlet	korlátozott	kiterjesztett	kiterjesztett + DAC
Kiterjesztett utasításkészlet és szolgáltatások		szorzó utasítások, nagyobb program-memóriákat kezelő utasítások	DMA, eseménykezelő-rendszer, titkosítás.

A választás - Atmel ATmega32

ATmega32L mikrokontroller, melynek megjelölésében a 32-es szám 32kB Flash memóriát jelent, az L betű a Low Power tulajdonságot, ebből kifolyólag alacsony feszültségen is képes üzemelni 2.7V - 5.5V között.

Ez esetben is a legfőbb információforrás a gyártó által kibocsátott adatlap (datasheet):

1. Adatlap - ATmega32 néhány főbb paraméterének felsorolása

- **High-performance, Low-power Atmel®AVR® 8-bit Microcontroller**

- **Advanced RISC Architecture**

- 131 Powerful Instructions – Most Single-clock Cycle Execution
- 32 × 8 General Purpose Working Registers
- Fully Static Operation
- Up to 16 MIPS Throughput at 16MHz
 - On-chip 2-cycle Multiplier

- **High Endurance Non-volatile Memory segments**

- 32Kbytes of In-System Self-programmable Flash program memory
- 1024Bytes EEPROM
- 2Kbytes Internal SRAM
- Write/Erase Cycles:
10,000 Flash/100,000 EEPROM
- Data retention:
20 years at 85°C/100 years at 25°C
- Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

- Programming Lock for Software Security

- **JTAG (IEEE 1149.1 Compliant) Interface**

- Boundary-scan Capabilities According to the JTAG Standard
- Extensive On-chip Debug Support
- Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

- **Peripheral Features**

- Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Real Time Counter with Separate Oscillator

- Four PWM Channels

- 8-channel, 10-bit ADC

8 Single-ended Channels

7 Differential Channels in TQFP Package Only

2 Differential Channels with Programmable Gain at 1x, 10x, or 200x

- Byte-oriented Two-wire Serial Interface

- Programmable Serial USART

- Master/Slave SPI Serial Interface

- Programmable Watchdog Timer with Separate On-chip Oscillator

- On-chip Analog Comparator

- **Special Microcontroller Features**

- Power-on Reset and Programmable Brown-out Detection

- Internal Calibrated RC Oscillator

- External and Internal Interrupt Sources

- Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby

- **I/O and Packages**

- 32 Programmable I/O Lines

- 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF

- **Operating Voltages**

- 2.7V - 5.5V for ATmega32L

- 4.5V - 5.5V for ATmega32

- **Speed Grades**

- 0 - 8MHz for ATmega32L

- 0 - 16MHz for ATmega32

- Power Consumption at 1MHz, 3V, 25°C

- Active: 1.1mA

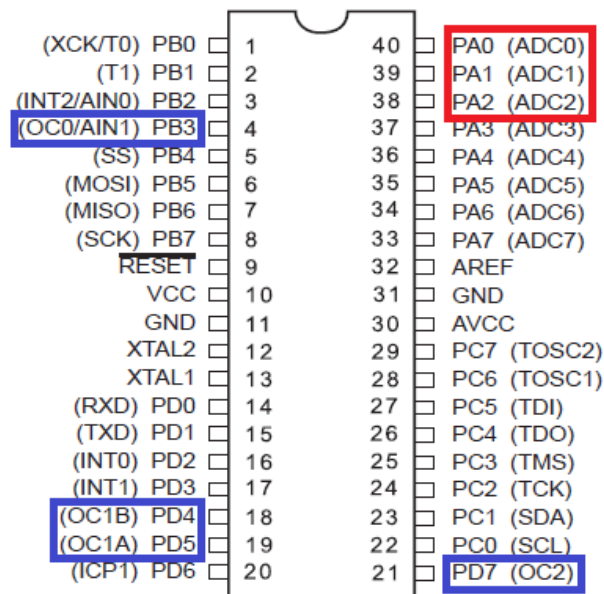
- Idle Mode: 0.35mA

- Power-down Mode: < 1µA

Forrás: Atmel: ATmega 32 datasheet [31].

Mint látható 4x8 láb használható az A, B, C, D be/kimeneti PORTok bekötésére (PORTA, PORTB, PORTC, PORTD). Mindegyik láb programozástól függően lehet be- vagy kimeneti PINx/PORTx lehet (pl. PINA/PA1,... PORTC/PC1,...). Ezt a PORThoz tartozó DDRx regiszterbe állíthatjuk be. Egyes lábak rendelkeznek egyéb funkcionalitással, mint az 1 ábrán pirossal jelölt A/D átalakítók, vagy a kézzel jelölt jelgenerátorok. Az PORTA A/D átalakítói lesznek használva az érzékelők lekérdezéséhez (ADC0, ADC1, ADC2). Ez esetben az A port első három PINjét az A/D átalakítóra kell

kapcsolni. Az LCD panel szintén igényel egy teljes portot. Terveink szerint a D port lesz erre felhasználva. A gombok és a kimentek a maradék portokat fogják használni, és még egy külső megszakítási vonalat, amely szintén a PORTD PD2, PD3 kivezetések extra funkcionalitása (INT0, INT1). Egyes lábak ilyen extra funkcionalitása látható az 1-es ábrán a zárójeles részben jelölve. Természetesen ezen szolgáltatásokat használva a lábak ki/be meneti funkcionalitása már nem használható.

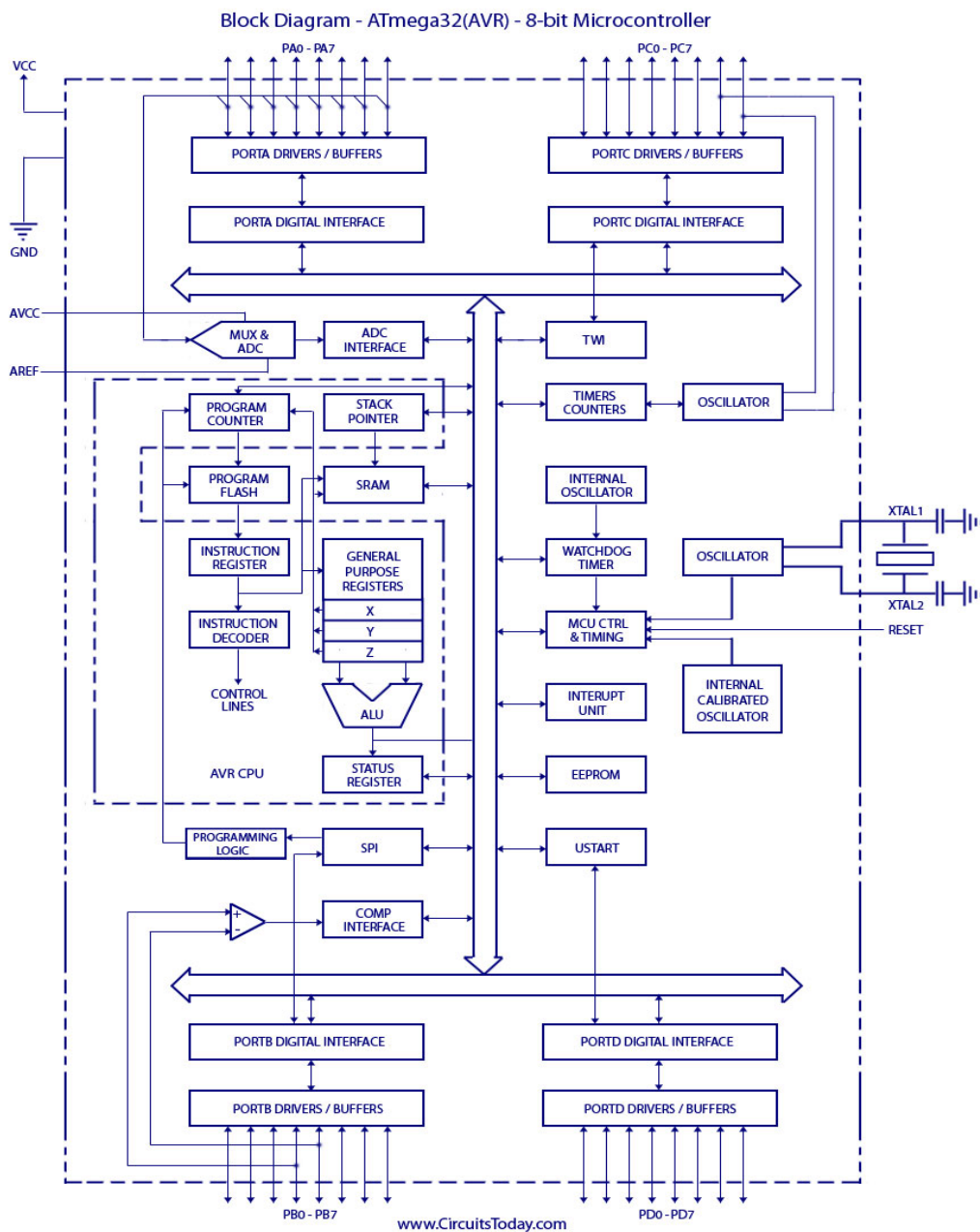


2. ábra: ATmega32 PDIP40 tokozású mikrovezérlő lábkiosztása

Forrás: Atmel: ATmega 32 datasheet [31].

Mikrokontrollerünk 8 csatornán biztosít 10 bites A/D konverziót. A konverziót az ADSC (ADC Start Conversion) bit beállításával indítjuk. Az átalakítás ideje alatt a bit beállítva marad, az átalakítás után pedig azonnal törlődik. Ha a konverzió közben csatornát váltunk, a konverzió olyankor is végbemegy. Az ADC másik üzemmódjában a hardver folyamatosan mintavételez és frissíti az ADC adat regiszterét. Ezt a folyamatos konverziót biztosító üzemmódot az ADCSRA regiszter ADFR bitjének beállításával állítjuk be. Az üzemmód beállítása még nem jelenti a konverzió kezdetét. Az első átalakítást az ADCSRA regiszter ADSC bitjének beállításával indítjuk el. A maximális felbontás eléréséhez az átalakítónak egy 50 és 200 kHz közötti frekvenciájú bemenő órajelre van szüksége. Ha 10 bitnél kisebb felbontással is megelégszünk, a bemenő frekvencia 200 kHz-nél nagyobb is lehet, így sűrűbb lesz a mintavételezés is. Az ADC modul tartalmaz egy előosztót, amely a megfelelő bemenő órajel előállításáért felel. Ezt bármilyen 100 kHz fölötti CPU-

frekvenciából elő tudja állítani. Az elő osztás mértékét az ADCSRA regiszter ADPS bitjeivel állítjuk be. Az elő osztó az ADC bekapcsolásakor – az ADEN bit beállításakor – kezd el számlálni. Amíg az ADEN be van állítva, folyamatosan számlál, amíg nincs beállítva, folyamatosan újraindul. Egy átalakítás elvégzése után az eredmény beíródik az ADC adat regiszterébe, az ADIF bit pedig beállítódik (azaz 1 lesz). Ha egyszeri átalakítást hajtottunk végre, az ADSC törlődik. Folyamatos átalakításkor azonban egy átalakítás befejezése után a következő azonnal elkezdődik, és az ADSC nem törlődik, beállítva marad.[3]



3. ábra: ATmega32 belső felépítésének blokk vázlata

Forrás: Atmel: ATmega 32 datasheet [31].

3.4.A környezet kiválasztása

Hasonlóan a processzorok kiválasztásához a programozási környezet kiválasztásánál is rengeteg lehetőség közül lehet választani. Legmegbízhatóbbnak maga a gyártó által kínált programozói felület kínálkozott, mely ingyenes voltán azonnal installálásra és kipróbálásra is került. Természetesen más környezetek is megfontolásra, installálás és kipróbálásra kerültek, ebben számunkra nagy előnyt főleg ingyenes mivoltuk jelentett. A program nyelvet többnyire a környezetek maguk meg is határozzák, így az általunk választott AVR Studio által biztosított C nyelv és Assembly nyelv volt adott.

3. Táblázat: Felsorolás fejlesztői környezetek és támogatott nyelvek

Gyártó	Fordító	Licenc	Platform	Assembler	BASIC	C	C++	Pascal	Egyéb
GNU	avr-as	GPL	Win32, Linux, Mac OS	Igen	–	–	–	–	–
GNU	avr-gcc	GPL	Win32, Linux, Mac OS	–	–	Igen	Igen	–	–
GNU	AVR-ADA	GPL	Win32, Linux	–	–	–	–	–	Ada
Atmel	AVR Studio	Nem	Win32	Igen	–	Igen	Igen	–	–
cadManiac.org	KontrollerLab	GPL	Linux	Igen	–	Igen	Igen	–	–
E-LAB Computers	AVRco	Nem	Win32	–	–	–	–	Igen	–
MCS Electronics	BASCOM	Nem	Win32	–	Dialektus	–	–	–	–
nettypes.de	mBasic	Nem	Win32	–	Igen	–	–	–	–
Till Harbaum	NanoVM	GPL	Win32, Linux	–	–	–	–	–	Java
Matthias Trute	amforth	GPL	Win32, Linux	–	–	–	–	–	Forth
?	Embedded Pascal for the AVR	share-ware	Win32	–	–	–	–	Igen	–
Dean W. Hall et al.	PyMite	GPL	Win32, Linux, Mac OS	–	–	–	–	–	Python
IAR	IAR Embedded Workbench for AVR	Nem	Win32	–	–	Igen	Igen	–	–
myAVR	myAVR Workpad	Nem	Win32	Igen	–	Igen	–	–	–
Gennady Gromov	Algorithm Builder	Nem	Win32	Igen	–	–	–	–	–
Jon A. Haugum, Tobias Weber	AVRA	GPL	független	Igen	–	–	–	–	bővített makrónyelv?
HP InfoTech	CodeVisionAVR	Nem	Win32	–	–	Igen	–	–	–
Jörg Wolfram	AVR-Chipbasic2	GPL	On Chip	–	Tiny-Basic	–	–	–	–
Mikroelektronika	MikroPascal for AVR	Nem	Win32	–	–	–	–	Igen	–
Thinker.it	Wiring/Arduino	GPL	Win32, Linux, MacOS	–	–	Igen	–	–	Bővített C-makro
ImageCraft	ICCV8 for AVR	Nem	Win32	Igen	–	Igen	–	–	CodeBlocks IDE

Forrás: Atmel AVR. 2014 [13].

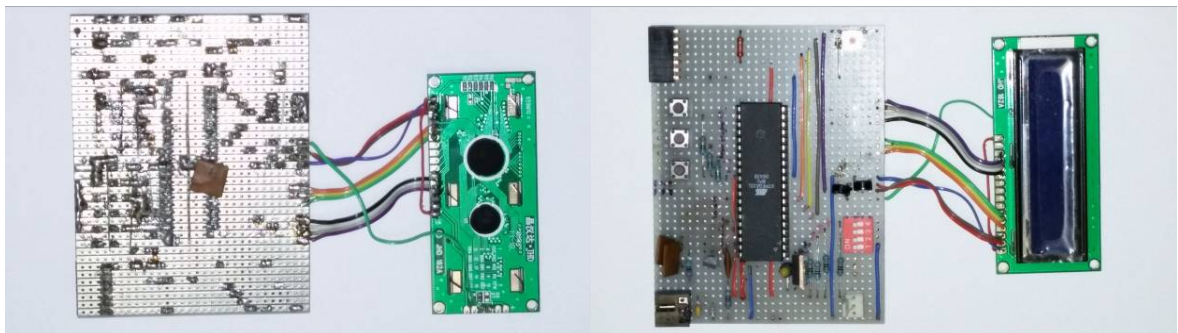
Választásunk az AVR Studio 6-ra esett, és ezen belül is a C nyelre. A C nyelvezete egyszerű, de mégis programozó közelebb, megértése egyszerűbb, mint a processzorhoz közelebb Assembly nyelv. Ennek ellenére nagyon tiszta, az interpreter számára könnyen egyszerűsíthető kódot fordít.

3.5. Nyomtatott áramköri lap

Az áramköri lap megtervezése az Eagle Layout Editor 6.5.0-ás verziójával készült, ami meghatározott méretig ingyenesen használható.

Szükségünk lesz:

- még néhány nyomógombra a menürendszer kezeléséhez és felhúzó ellenállásokra, melyek segítségével tartja a logikai bemenetet magas szinten, hogy csatlakozás hiányában ne kerüljön határozatlan állapotban a bemenet.
- diódákra a nyomógombok lábainak a megszakításra kötéséhez. Néhány LED diódra a kijelzéshez, meghajtásukhoz szintén kell ellenállás is. Az ADC-inputok lábainak filtrálására M1-es keramikus kondenzátorok szükségesek.
- csatlakozókra, jack tápfeszültség csatlakozó, programozó csatlakozó, ventilátor kimenet csatlakozó. LCD panel csatlakozó és persze szükségünk lesz egy LCD panelre is a mért adatok megjelenítéséhez. Az LCD panelek általában a Hitachi HD44780 lcd panel klónok, így ez majdnem egy ipar szabvánnyá nőtte ki magát. Ezért az LCD panel kiválasztása nem okozott különösebb gondot. Több fajta is felhasználásra került a munka közben. Végleges felhasználásra végül JHD162A típus került.



4. ábra: Áramköri lap a gyors fejlesztéshez

A képen a nyomtatott áramköri lap még fejlesztés alatt áll. Ebben az állapotában egy próbapanelen vannak összerakva az alkatrészek, amely a tesztelés fázisában megfelel. A cél az, hogy gyorsan lehessen tesztelni a kód és a hardver működését. A végleges áramkör csak hosszas tervezési és tesztelési folyamat végállomása, ami esetleges elrontott kísérletek és több próbapanel elkészítését igénylik. A tervezett áramköri lap egyoldalas. Az áramköri elemek összekötésének vázlati rajza, és a nyomtatott áramköri lap terve a 2. mellékletben található.

3.6. Programozási alapismeretek

Alap adattípusok

1. Char: 1 byte
2. Short int: 2 byte
3. Int: 4 byte
4. Long Int: 8 byte
5. Float: 32 bites, IEEE 754 egyszeres pontosság
6. Double: 64 bites, IEEE 754 dupla pontosság
7. Long Double: 128 bites, IEEE 754 négyszeres pontosság

Típus módosítók:

1. Auto: a változó csak abban a blokkban látszik, ahol definiáltuk a kezdő érték adással, használata nem kötelező, mert minden változó alapértelmezetten így jön létre.
2. Const: a változónak csak definiáláskor lehet értéket adni, később nem lehet megváltoztatni.
3. Extern: másik forrásfájlban használt változó, vagy függvény
4. Register: a regiszterben tárolt változó lesz (ha lehetséges)
5. Signed/Unsigned: kényszerített előjeles/nélküli, csak char típushoz.
6. Static: a változó funkcióhívások közben is megőrzi értékét, alapértelmezetten a globális változó ilyen.
7. Volatile: hatására a változó a memóriában fog tárolódni.

Operátorok

Alap műveletek:

+ plusz, - mínusz, * szorzás, / osztás, % osztásmaradék

Bitműveletek:

~ bit-szintű NEM operátor

& bit-szintű ÉS operátor

| bit-szintű VAGY operátor

^ bit-szintű KIZÁRÓ-VAGY (XOR) operátor

« biteltoló (shift) operátor (balra, MSB felé – 0 értékű biteket tölt be az LSB felől)

Logikai operátorok:

! logikai nem (not)

== egyenlő

!= nem egyenlő

|| logikai vagy

&& logikai és

< logikai kisebb

> logikai nagyobb

Értékadó operátorok

= legyen egyenlő

+= változó értéke legyen megnövelve

-= változó értéke legyen csökkentve

*= változó értéke legyen szorozva

/= változó értéke legyen osztva

|= változó értéke legyen bitenkénti vagy kapcsolatba hozva maszkal

&= változó értéke legyen bitenkénti és kapcsolatba hozva maszkal

^= változó értéke legyen bitenkénti xor kapcsolatba hozva maszkal

++, inc, függően az elhelyezéstől ++a művelet előtti, a++ művelet utáni

--, dec, függően az elhelyezéstől --a művelet előtti, a-- művelet utáni

Egész számok megadása

Egész számokat tízes, tizenhatos, nyolcas és bináris számrendszerben adhatunk meg a következő képpen:

Int a=13 tízes számrendszerben történő megadás

Int a=013 oktális számrendszerben történő megadás

Int a=0x13 hexa számrendszerben történő megadás

Int a=0b1101 bináris számrendszerben történő megadás

[8]

Eszközők a változók bitjeinek manipulálásához:

egy bit/bitek értékének a megállapítása,

egy bit/bitek beállítása 1-be,

egy bit/bitek nullázása.

Minden bitmanipuláló művelet alapja

1. maszk előállítása: biztosítja, hogy az adott logikai művelet csak a kérdéses bitekre legyen hatással,
2. logikai művelet: elvégzi a szükséges bitmanipulációt.

Bit értékének a megállapítása

A bitvizsgálat alapl művelete: ÉS.

Ha egy ismeretlen bitet 1-gyel ÉS kapcsolatba hozunk, akkor az eredmény pontosan a kérdéses bit lesz. Ha egy bitet 0-val hozunk ÉS kapcsolatba, az eredmény 0 lesz. Tehát a változónk kérdéses bitjének/bitjeinek a helyére a maszkba 1-est, a többi helyre 0-t kell írni. Így az eredmény:

0, ha a kérdéses bit(ek) 0 volt(ak),

sum(2i), ha a kérdéses i-edik bit(ek) 1-es értékű(ek) volt(ak).

Példa: vizsgáljuk a PINC változó 7. bitjét. A maszk a 7. bitjére 1-est kell írni – használhatjuk a biteltoló operátort, vagy létrehozunk egy bináris maszkot:

```
maszk = 1 << 6; maszk = 0b01000000>;  
eredmeny = PINC & maszk;  
if (eredmeny != 0)  
if (!(PINC & 0b01000000))
```

Bit beállítása 1-be

A bitbeállítás alapl művelete: VAGY

Ha egy ismeretlen bitet 0-val VAGY kapcsolatba hozunk, akkor az eredmény pontosan a kérdéses bit lesz. Ha egy bitet 1-gyel hozunk VAGY kapcsolatba, az eredmény 1 lesz. Tehát a változónk kérdéses bitjének/bitjeinek a helyére 1-est, a többi helyre 0-t kell írni a maszkba, majd a VAGY művelet eredményét vissza kell írni a változóba. Így csak azok a bitek változnak, amelyek helyén a maszkban 1 áll.

Példa: állítsuk a DDRB változó PB0 bitjét 1-be:

```
maszk = 1 << PB0;  
DDRB = DDRB | maszk;  
Ugyanez egy sorba:  
DDRB |= (1<<PB0);
```

Bit beállítása 0-ba

A bit törlésének alpművelete: ÉS.

A feladat: a kérdéses bitet ÉS kapcsolatba kell hozni 0-val, a többi 1-gyel és az eredményt vissza kell írni a változóba.

Csak hogy a biteltoló operátor mindig 0-t helyez be, ezért a trükk: a szükséges bitminta invertáltját állítjuk elő, majd negálunk.

Példa: állítsuk a DDRD változó PD3 bitjét 0-ba:

```
maszk = ~(1 << PD3);  
DDRD = DDRD & maszk;  
Ugyanez egy sorba:  
PORTD &=~(1<<PD3);
```

Az egyszerű program felépítése:

```
#include <...>           // A szükséges fejléc-állományok beszerkesztése.  
SIGNAL (SIG_...) {...} // Megszakítás-kezelő rutin(ok).  
void init (void) {...} // Inicializáló függvény.  
int main() {             // A főprogram  
    init ();             // Az inicializáció.  
    while (1) {...}      // Végtelen ciklus(!) – a munka a  
                           kikapcsolásig tart (egyszerű esetekben).  
    return 0;            // Soha végre nem hajtott return a fordító kedvéért.  
}
```

A fejlett energiatakarékos program felépítése

Az egyszerű program körkörös futást úgynevezett pooling-ot használ, hogy minden egyes program rész megfelelő processzor időt kapjon, melyet jól beállított késleltető delay() függvényekkel lehet többé-kevésbé megfelelően időzítetté alakítani. Ezzel szemben egy korszerű rendszer pontos időzítéseket kíván, reakció ideje kritikus műveleteknél elengedhetetlen, de viszont nincs egyéb dolga a rendszernek, akkor ne dolgozzon feleslegesen. Így ha engedélyezzük kihasználható a processzorba épített energiatakarékosság funkció, mely alacsony áramfelvételi állapotba kapcsolja a processzort, ha éppen nem futtat program kódot.

```

#include <...>                // A szükséges fejléc-állományok beszerkesztése.
ISR (TIMERi_...) { ... // Megszakítás-kezelő rutin(ok).
    Ti=true;                // A túl hosszú rutin, egy jelző engedélyezésével
    }                      // folytatja majd a munkát a főprogramban.
ISR (INTi_...) { ... // Megszakítás-kezelő rutin(ok).
    Ii=true; }
void init (void) { ... } // Inicializáló függvény.
int main() {                // A főprogram
    init ();                // Az inicializáció.
    while (1) { ... // Végtelen ciklus(!) itt is, de csak egyszer fut le
        if(Ti) { ... // Megszakítás-kezelő rutin folytatása.
            Ti=false; // jelző visszaállítása.
        }
        if(Ii) { ...
            Ii=false; }
        sleep_mode(); // Ha a feladatok befejeződtek, alvó állapot(!).
    } // Végtelen ciklus(!) – itt is kikapcsolásig tart.
    return 0; // Soha végre nem hajtott return a fordító kedvéért.
}

```

Kritikus szakaszoknál letilthatóak a megszakítások cli(), így megakadályozható a megszakítások végtelen számú rekurzióba hívása. Ezért is fontos, hogy a megszakítás kezelő rutinok kódja legyen rövid, velős, ha több feladatot szeretnénk elvégeztetni, azt tegyük a főprogramban. A munka végeztével sei() utasítással újra engedélyezzük a megszakításokat.

Az inicializáló függvény

Hasonló módon az inicializáló függvényünk, ami mindenképp szükséges a helyes működés beállításához, hogy ne legyen megszakítva, az elejére egy cli() a végére egy sei() utasítást teszünk. Az inicializáló függvényben állítjuk be a mikrokontrollerünk portjait a megfelelő BE/KI meneti állapotba. Ugyanúgy itt állíthatjuk be a megszakítások paramétereit, és engedélyezésüket.

Processzorunk lábait be kell állítani BE- vagy KI- menet, vagy éppen egy speciális funkció használatára, ilyen lehet ACD bemenet, PWM kimenet, Counter bemenet.

Számláló – counter inicializálása

Mivel a számláló nagyon sokat használt modul a vezérlésben, általában több is van belőle. Ennek beállításait is itt kell megadni, mint az előosztó: a mikrovezérlő órajelét szabályozható értékkel elosztja, alapérték: megszakítást tud adni egy adott érték elérésekor / túlsordulásakor. Az Atmega32-ben a 0-s számláló egy egyszerű 8-bites időzítő.

Négy konfigurációs regiszter kapcsolódik hozzá:

TCCR0 – Timer/Counter Control Register 0: üzemmód választás

TCNT0 – Timer/Counter Register: a számláló aktuális értéke

TIMSK – Timer/counter Interrupt Mask Register: megszakítások engedélyezése

TIFR – Timer/counter Interrupt Flag Register: megszakítás flagek

Számunkra a TCCR0 és a TIMSK lesznek fontosak. Ezért ezeket részletesebben vizsgáljuk meg.

4. Táblázat: TCCR0 regiszter bitjei

	7	6	5	4	3	2	1	0
TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Forrás: ATmega 32 datasheet [31]

5. Táblázat: TIMSK regiszter bitjei

	7	6	5	4	3	2	1	0
TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

Forrás: ATmega 32 datasheet [31]

6. Táblázat: A TCCR0 0-2 bitjeinek (CS00-CS02) jelentése:

CS02	CS01	CS00	Jelentés
0	0	0	A számláló inaktív.
0	0	1	Nincs előosztás.
0	1	0	8-as előosztás.
0	1	1	64-es előosztás.
1	0	0	256-os előosztás.
1	0	1	1024-es előosztás.
1	1	0	Külso órajel (T0 láb), le futó él
1	1	1	Külso órajel (T0 láb), fel futó él

Forrás: ATmega 32 datasheet [31]

Mivel a legalacsonyabb belső órajel 1 MHz, a legnagyobb előosztást érdemes használni, így a bitek értéke: 101 lesz, ami hexadecimálisan 0x05. Megszakításokat is szeretnénk kapni, ehhez a TIMSK regiszter legalsó bitjét kell beállítani:

TOIE0: Timer/Counter0 Overflow Interrupt Enable

Ennek hatására, amikor a számláló túlsordul (a 255 érték elérése után): a TIFR regiszter 0. bitje 1-es értéket kap (TOV0 – Timer/Counter0 Overflow Flag), megszakítást kapunk: (SIG_OVERFLOW0).

A lekérdezés frekvenciája így: az 1 MHz-es órajelet 1024-gyel előosztjuk, a számláló még 256-al oszt, hiszen minden 256. órajelnél kapunk megszakítást, tehát $1\text{MHz} / 1024 / 256 = \uparrow 2\text{ Hz}$.

ADC inicializálás

Két konfigurációs regiszter kapcsolódik hozzá: ADMUX és ADCSRA.

7. Táblázat: ADMUX ADC csatornát és referencia feszültséget beállító regiszter.

	7	6	5	4	3	2	1	0
ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Forrás: ATmega 32 datasheet [31]

8. Táblázat: Ebből a 6 és 7 bit a referencia feszültséget határozza meg

REFS1	REFS0	Referencia feszültség választás
0	0	AREF, Internal Vref turned off
0	1	AVCC külső kapacitással az AREF lábán
1	0	Nem használt
1	1	Belső 2.56V-os refer.feszültség külső kap. az AREF lábán

Forrás: ATmega 32 datasheet [31]

9. Táblázat: A Mux0:2 bitek fogják meghatározni a szükséges ADC csatornát

MUX4	MUX3	MUX2	MUX1	MUX0	Single Ended	Positive Different	Negative Different	Input Gain
0	0	0	0	0	ADC0	N/A		
0	0	0	0	1	ADC1			
0	0	0	1	0	ADC2			
0	0	0	1	1	ADC3			
0	0	1	0	0	ADC4			
0	0	1	0	1	ADC5			
0	0	1	1	0	ADC6			
0	0	1	1	1	ADC7			
0	1	x	x	x	N/A	Differenciális és erősített AD k.		
1	x	x	x	x				
1	1	1	1	0	1.22V			
1	1	1	1	1	0V			

Forrás: ATmega 32 datasheet [31]

ADLAR: ADC adat regisztereiben, ADCH és ADCL, a tárolt értéket 0 esetén jobbra, vagy 1 esetén balra rendezi.

10. Táblázat: Az ADCSRA regiszter bitjei

	7	6	5	4	3	2	1	0
ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Forrás: ATmega 32 datasheet [31]

ADEN: a bit 1-esre állításával bekapcsoljuk az ADC modult. 0-ki kapcsolja.

ADSC: a bit 1-esre állításával elindítjuk az AD konverziót. Olvasása esetén 1-es érték mutatja, hogy az ADC konverzió tart, amint befejeződik, értéke 0-ra változik. 0 írása a bitbe nem okoz változást.

ADATE: AD auto trigger enable. Az SFIOR ADTS bitje által beállított trigger jel felfutó élére indítja az AD konverziót.

ADIF: AD interrupt flag jelzi, hogy kész a konverzió és az adatok készen állnak, indítható az ADC Conversion Complete Interrupt. 1-es írása törli a jelzést.

ADIE: AD interrupt enable bit és az SREG I-bitje 1-esbe van állítva, akkor engedélyezve van az ADC Conversion Complete Interrupt.

ADPS2:0 az AD prescaler előosztót beállító bitek.

11. Táblázat: Az ADPS0:2 bitek jelentés

ADPS2	ADPS1	ADPS0	Osztó
0	0	0	2-es előosztás.
0	0	1	2-es előosztás.
0	1	0	4-es előosztás.
0	1	1	8-as előosztás.
1	0	0	16-os előosztás.
1	0	1	32-es előosztás.
1	1	0	64-es előosztás.
1	1	1	128-as előosztás.

Forrás: ATmega 32 datasheet [31]

ADC Data Regiszter – ADCL és ADCH

Ebből a regiszterből olvashatjuk ki az ADC eredményeket. Az ADLAR bit értékétől függően jobbra, vagy balra rendezett formában.

12. Táblázat: ADCL és ADCH ADC adat regiszterei ADLAR=0 esetén

	15	14	13	12	11	10	9	8
ADCH	–	–	–	–	–	–	ADC9	ADC8
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

Forrás: ATmega 32 datasheet [31]

13. Táblázat: ADCL és ADCH ADC adat regiszterei ADLAR=1 esetén

	15	14	13	12	11	10	9	8
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	–	–	–	–	–	–

Forrás: ATmega 32 datasheet [31]

Külső megszakítás – interrupt

14. Táblázat

	7	6	5	4	3	2	1	0
GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE

Forrás: ATmega 32 datasheet [31]

A GICR regiszter 5-7 bitje az INT0, INT1, INT2 engedélyezi a megfelelő külső megszakítást. A bitbe 1-est írva engedélyezzük a megszakítást, 0 a tiltást eredményezi.

15. Táblázat

	7	6	5	4	3	2	1	0
MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00

Forrás: ATmega 32 datasheet [31]

Az MCUCR regiszter alsó 4 bitje a számunkra érdekes, ezek határozzák meg, hogy a külső megszakítások, amely jelváltozás állapotban történjenek meg.

16. Táblázat

ISC01	ISC00	Leírás
ISC11	ISC10	
0	0	Az INT0 láb alacsony szintje megszakítást vált ki.
0	1	Az INT0 lábon történő logikai változás megszakítást vált ki.
1	0	Az INT0 lábon a jel lefutó éle megszakítást vált ki.
1	1	Az INT0 lábon a jel felfutó éle megszakítást vált ki.

Forrás: ATmega 32 datasheet [31]

3.7. Program kód

A mikrokontrollerek programja általában strukturált szerkezetű. Léteznek objektum orientált könyvtárak, melyek viszont megnövelik a fordított kód méretét, ami nem szerencsés korlátozott program memória esetén (esetünkben 32kB). Előre megírt könyvtárak segítik munkánkat, melyek egyszerűbben érthető változókká (az Assemblyhez hasonlóan mnemonic szavakká) fordítja a processzorunk regisztereit és állapotregisztereit. Ilyen könyvtárak az AVR GCC az io.h, interrupt.h, delay.h, és az lcd.h. Az egyes könyvtárak betöltése után, inicializálnunk kell a mikrokontrollerünk be- és kimeneteit, megszakításait, az ADC-t. Majd inicializálás után írhatjuk is a kódot egy úgynevezett LOOP FOREVER végtelen ciklusban, mivel a mikrokontrollerünknek ezt a feladatot kell végeznie egészen a kikapcsolásig. A programkód elkészítését megelőzően elkészítettünk egy egyszerűsített programfutás szerkezeti folyamatábrát, ez nagymértékben leegyszerűsíti programunk megírását.

Ez a programkód pooling szerűen működik, mely kiegészült a az időzítők, a megszakítások által nyújtott pontosabb időzítésekkel. A fő program két fő részre választható szét. Az első rész az alapvető funkcionalitást biztosítja. Ezek az alapvető funkciók, melyeknek mindig végre kell hajtódniuk, akkor is ha a menüben leszünk:

1. A mérések végrehajtása.

A mérésekhez létre lett hozva egy függvény, amely paraméterként a lekérdezendő ADC csatornát várja, és eredményként az ADC mérés eredményét adja vissza. Amíg a mérés tart várakozik. Ez egyszeri mérés esetén 13 gépi ciklus, méréseink pontosabbá tétel végett 8 mérést végzünk és ezek átlagát adjuk vissza eredménynek. Ha processzorunk sebessége 1MHz, akkor $8 \cdot 13 = 104$ + egyéb műveletek (összeadás, átlagszámítás, ciklus) megközelítőleg 125, $1000\text{kHz} / 125 = 8\text{kHz}$, $1/8000 = 0,125\text{ms}$ lesz egy mérés ideje.

2. A mérések eredményeinek kiértékelése, és -

3. A kimenetek vezérlése az eredmények alapján.

A mérések kiértékelések és a kapcsolások létrehozása:

```
if (adc_result0 >= HIGH) PORTD |= (1<<PINBx);  
if (adc_result0 <= LOW)  PORTD &= ~(1<<PINBx);
```

A második rész a képernyőn való megjelenítés is kettő részből áll:

Ennek eldöntését a program első verziójában szintén pooling szerűen minden egyes mérésnél, vagyis a program első részének végén lekérdeztük a gombok által használt PINek állapotát. A programunk második verziója kiegészült a gombok által létrehozott külső megszakításokkal, ezeket hardveresen diódákkal kötöttünk an INT1 lábra. Melyek így azonnali reakcióval beállítanak egy jelzőt, miszerint történt gombnyomás, és így nem történhet meg az ami a poolingos lekérdezéses módszernél, hogy véletlenül lemaradnánk a megnyomott gomb érzékeléséről. Ugyanezzel a módszerrel tudjuk ki védeni a megnyomott gomb többszöri érzékelését (pergés probléma), mely a gombok rugós szerkezetéből adódóan a kapcsolás után még néhány ms-ig nem állapodik meg és egy bizonytalan állapotot okoz. A megszakításunk meghívásakor, a gombnyomás első érzékelése után, letiltjuk a megszakításunkat a pergés megszűnéséig (200 ms) is, így akkor nem érzékeli a hamis kapcsolásokat.

1. A mérések eredményeinek megjelenítése – alap esetben.

A megjelenítés rész, ha volt gombnyomás a menüt jeleníti meg, ellenkező esetben a mérések eredményeit, amelyet nevezhetnünk az alap eset befejeződésnek. Ezután már csak a megfelelő várakozási idő után a végtelen ciklusunk előről indítja a főprogramunk.

A mérések eredményeinek megjelenítése egy egész rövid rész, melyben a megkapott eredményeket az ITOA függvénnyel karakterekké alakítja és az lcd.h könyvtár lcd_gotoxy és az lcd_putchar funkciók segítségével a megfelelő helyre kiírja.

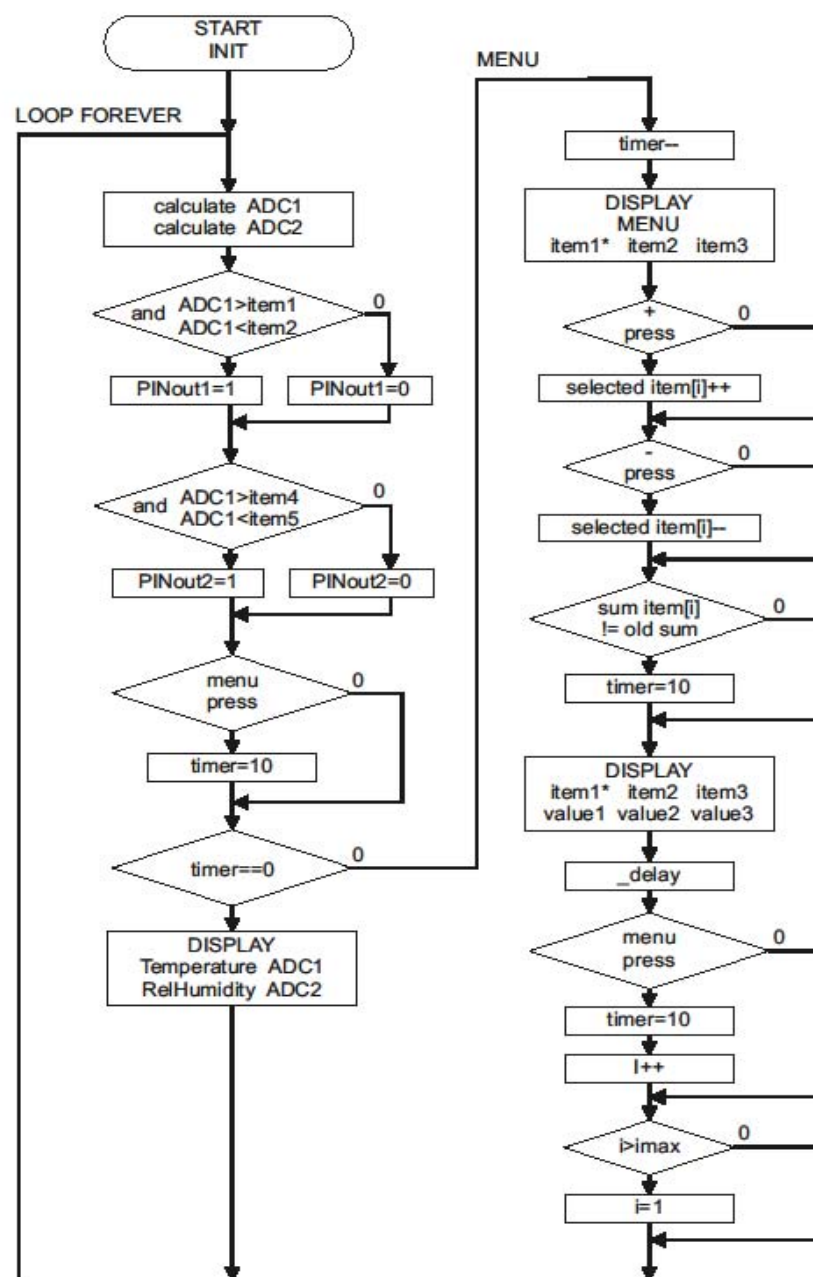
2. A menü megjelenítése – ha valamely gombot megnyomtuk.

Ha a menübe léptünk, hasonlóan az LCD.h könyvtár funkciói segítségével megjelenítjük a megfelelő helyeken a menü aktuális szintjén lévő elemeket. Az aktuálisan állítható elemet csillaggal jelzi. Ezek közül a menü gomb további nyomogatásával választhatunk.

"TLO* THI TEMP "	Menü 1
"TLO THI* TEMP "	Menü 2
"TLO THI TEMP*"	Menü 3
"HLO* HHI HUMI "	Menü 4
"HLO HHI* HUMI "	Menü 5
"HLO HHI HUMI*"	Menü 6

Lásd még melléklet 3.

Ha a megfelelő paraméteren állunk az értéket a + és – gombok használatával csökkenthetjük vagy növelhetjük. Ekkor a paraméter alatt lévő érték interaktívan megváltozik. Ha beállítottuk a megfelelő paramétereket, és nem kívánunk továbbiakban állítani, az utolsó gomb megnyomásától számítva egy számláló elkezd 30-ról visszaszámlálni. Ez nagyságrendileg 15 mp-nek felel meg. Ha a továbbiakban sem nyomunk meg semmilyen gombot, és a számláló eléri a 0-t, akkor a rendszer a következő ciklusban ismét a főprogram alap megjelenítő részébe lép, és a mérések eredményeit fogja mutatni.



5. ábra: Programszerkezet folyamatábrája

3.8. Tesztelés

A fejlesztés során végig szükségünk van a tesztelésre, mely rámutathat rendszerünk hiányosságaira, s így azonnali javítást tesz lehetővé. Az Atmel fejlesztői rendszere együtt az AVR mikrokontrollerekkel lehetőséget biztosít futás közbeni hibakeresésre, programunkat futása közben bárhol megállíthatjuk, léptethetjük, és figyelemmel kísérhetjük regisztereik és változóik értékét. A tesztelést így nevezhetjük a fejlesztési folyamat elengedhetetlen részének, ami a kifejlesztett eszköz további élete folyamán sem áll meg, sőt a valós élet, a reális alkalmazás rámutathat fel nem fedezett rejtett hibákra, melyeket orvosolni kell. Leszegezhetjük a fejlődés soha nem áll meg.

Tesztelés során derült fény olyan problémákra, mint az AD konverzió 1 bit pontatlansága, vagy zaj megjelenése filtráló kondenzátorok nélkül, vagy túl sok kondenzátor használata esetén a rendszer megnövekedett kapacitása, vagy a nyomógomb kapcsolók pergésének problémájára.

Az AD konverzió pontosságának növeléséhez a konverzió többszöri megismétlése és eredményeinek átlagolásával jutottunk. Esetünkben 8 mérés eredményének összeadása, és az összeg elosztása 8 a megoldás.

A zaj szűrésére az AD bemeneti láb és a föld közé M1-es az-az 100nF kapacitású keramikus kondenzátorok bekötésére van szükség. A processzor tápfeszültségét és földjét párhuzamosan szintúgy M1-es keramikus kondenzátorral kell ellátni, megóvva így processzorunkat a felesleges környezeti zajtól.

A pergő kapcsolók problémája hardveres megoldással szintén kondenzátorokat kívánna, de nem javasolt az áramkörünket telerakni kondenzátorokkal, mert megnövekedett impedanciája bizonytalan működését eredményezheti. Létezik egy szoftveres megoldása is a problémának, melyet alkalmazunk is rendszerünkben. Ez az első beérkező interrupt impulzus (ami a gombnyomás) után kis időre (esetünkben 50ns) letiltja az interruptok használatát, így a további impulzusok lecsengéséig nem érzékeli a beérkező jeleket.

4. Összegzés

A munkám elején megfogalmazott problémát az általunk épített rendszer sikeresen megoldja, amely alapján biztosítani kell a mini üvegház belső klímáját, és egyben biztosítani a hőmérséklet és páratartalom megfelelő szinten tartását, és ezen fizikai mértékek kijelzését.

Túlmelegedés esetén kapcsolja a szellőztetést, példánkban ez egy kis ventilátor, ami a környezet friss, hűvösebb levegőjét juttatja mini üvegházunkba, lehűlés esetén kapcsolja az izzót, amely által termelt hő felmelegíti. Rendszerünk lehetőséget nyújt a menü rendszeren keresztül ezen berendezések be és kikapcsolási értékeinek változtatására, így biztosítva az azonnali beavatkozás lehetőségét. A menün keresztül lehetőségünk van még a kalibrációs értékek beállítására is, ha rendszerünk valamely oknál kifolyólag elállítódna.

A rendszer továbbfejlesztése egy további munka keretén belül valósulna meg. Végleges nyomtatott áramköri lapon összeszerelt rendszer a környezeti hatásoktól megvédendő vízhatlan tokozásba lenne építve, univerzális csatlakozókkal lenne ellátva több különböző külső berendezés irányítására, usb-rs232 kommunikációval kiegészítve, a számítógéphez csatlakoztatás megoldására. A számítógéphez csatlakoztatott rendszer jelezné a pillanatnyi állapotot, az így megszerzett valós idejű adatokat elraktározva, grafikonnal ábrázolva mutatná egy meghatározott időtartam alatt történt változás függvényét, összehasonlítást biztosítana egyes periodikus ciklusok között, mint nappal-éjszaka, a hét napjai, hónapok, vagy más kritériumok szerinti elosztásban.

Befejezés

A dolgozatban kitűzött cél eléréséhez elméleti tanulmányokat végeztünk, majd az elnyert tudás alapján igyekeztünk a gyakorlatban kivitelezni egy olyan általunk megtervezett rendszert, amely használható és felhasználható lesz.

A dolgozatot öt fejezetre osztottuk fel, majd azokat témafüggően alfejezetek követik. A mellékelt ábrákat és táblázatokat jegyzékbe szedtük. Ezek az illusztrációk a dolgozat szerves részét képezik, mivel indoklásul szolgálnak az egyes gyakorlati lépések sorozatához.

A célkitűzés bizonyítása az általunk szerkesztett modell működőképessége volt.

A dolgozat a fizikai mértékegységek elméleti és gyakorlati felhasználásának lehetőségeivel foglalkozott, kísérletet tett a felvetett kérdések válaszainak bizonyítására és a mikrokontroller által irányított rendszerek beüzemeltetésére, ipari alkalmazására. Továbbá foglalkozott a mikrokontrollerek és programozásuk megismertetésével, programozási környezetük kiválasztásával, belőlük fejleszthető eszközök megvalósításával és egy minta rendszer megépítésével.

A mai virtuális világban az igények kielégítésére egyre gyakrabban használják a mikrokontrollereket. Megismerésük, fejlesztésük segít minket a karbantartásuknál, alkalmazásuknál is.

Szakirodalom

1. Ajtonyi, I.; Gyuricza, I.: *Programozható irányítóberendezések, hálózatok és rendszerek*. Budapest: Műszaki könyvkiadó. 2002. 500 s. ISBN 963 16 1897 8
2. Borsi, I. N.: *Mikrokontrollerek az informatika oktatásában* (Szakdolgozat). Debrecen: Debreceni Egyetem. 2007. 100 s.
3. Juhász, T.: *Beágyazott mérő- és szabályzórendszerek* (Diplomamunka). Debrecen: Debreceni Egyetem. 2007. 49 s.
4. Matoušek, D.: *Práce s mikrokontroléry*. Praha: BEN. 2006. 320 s. ISBN 80-7300-174-8
5. Nagy, G.: *Mikrovezérlők programozása* (Jegyzet). Budapest: BME. 2013. 33 s.
6. Pardue, J.: *C Programming for Microcontrollers*. Knoxville: Smiley Micros, 2005. 300 s. ISBN 0-9766822-0-6
7. Raffai, M.: *Az informatika fél évszázada*. Budapest: Springer. 1997. 422 s. ISBN 963 8455 61 6
8. Ruzsinszki, G.: *Programozható elektronikák*. [online]. 2013.3.8. 598 s. [cit. 2014.3.11.]. <http://www.scribd.com/doc/211374635/Programozhato-elektronikak>
9. Reh, B.: *An Introduction to programming an Atmega microcontroller*. [online]. 2013. 25 s. [cit. 2014. 3.11.]. <http://ornella.iwr.uni-heidelberg.de/ROBOTICSLAB/DEVELOPMENT/DOWNLOAD/atmega-tutorial.pdf>
10. Rodek, L.; Diós, G.: *Assembly Programozás* (Jegyzet). 138 s.
11. Váňa, V.: *Mikrokontroléry Atmel AVR – programová v jazyce C*. Praha: BEN. 2009. 216 s. ISBN 80-7300-102-0

WWW források:

12. *Arduino mikrokontroller programozása*. [online] [cit. 2014-5-11] http://logout.hu/cikk/arduino_mikrokontroller_programozasa/nyomtatobarat/teljes.html
13. *Atmel AVR*. [online] [cit. 2014-5-11] http://hu.wikipedia.org/wiki/Atmel_AVR
14. *Atmel mikrokontroller kártya*. [online] [cit. 2014-5-11] <http://members.chello.hu/bial.bela/Munkaim/munkaim.htm>
15. *AVR Libc Home Page*. [online] [cit. 2014-5-11] <http://www.nongnu.org/avr-libc/>

16. *DigiThermo 0-100.0 °C*. [online] [cit. 2014-5-11]
<http://chaokhun.kmitl.ac.th/~kswichit/thermo/digithermo.htm>
17. *DS1820RS*. [online] [cit. 2014-5-11]
<http://mikroklub.hu/htm/micros.htm>
18. *Digitális óra, vezetékek nélküli hőmérővel*. [online]
http://www.hobbielektronika.hu/cikkek/digitalis_ora_vezetek_nelkuli_homerovel.html
19. *Egyszerű hőmérő*. [online] [cit. 2014-5-11]
http://www.hobbielektronika.hu/cikkek/egyszeru_homero.html?pg=1
20. *Ismerkedés a PIC18 mikrovezérlővel*. [online] [cit. 2014-5-11]
<http://esca.atomki.hu/PICula/i2c.html>
21. *Mikrovezérlő*. [online] [cit. 2014-5-11]
<http://hu.wikipedia.org/wiki/Mikrovezérlő> 2014. március 11.
22. *Mikrovezérlők története*. [online] [cit. 2014-5-11]
<http://plc.mechatronika.hu/mikrovez/tortenet.htm> 2014. március 11.
23. *Parallax P8X32A Propeller Demo Board*. [online] [cit. 2014-5-11]
<http://www.puzsar.hu/propeller/index.html>
24. *Pontos LCD hőmérő MSP430-al*. [online] [cit. 2014-5-11]
http://www.hobbielektronika.hu/cikkek/pontos_lcd_homero_msp430-al.html?pg=1
25. *Raspberry Pi*. [online] [cit. 2014-5-11]
http://hu.wikipedia.org/wiki/Raspberry_Pi 2014. március 11.
26. *Rf kommunikációs hőmérő*. [online] [cit. 2014-5-11]
http://electronics-diy.com/electronic_schematic.php?id=725
27. *USB hőmérő mérésleírás*. [online] [cit. 2014-5-11]
http://fizipedia.bme.hu/index.php/USB_homero_meresleiras#Feladatok
28. *Using a Thermocouple*. [online] [cit. 2014-5-11]
<https://learn.adafruit.com/thermocouple/using-a-thermocouple>
29. *WILL-I - Robot, AVR mikrovezérlővel*. [online] [cit. 2014-5-11]
http://www.hobbielektronika.hu/cikkek/will-i_epitese_avagy_nullarol_a_robotokig_-_avr_mikrovezerlok.html?pg=1
30. *11 zónás öntözésvezérlő ATmega1284P-vel, RTC-vel, 1-vezetékes hőmérsékletérzékelőkkel, talajnedvesség-érzékelővel*. [online] [cit. 2014-5-11]
http://www.hobbielektronika.hu/cikkek/11_zonas_ontozesvezerlo_atmega1284p-vel_rtc-vel_1-vezetekes_homersekleterzekelokkal_talajnedvesség-erzekelovel.html?pg=1

Alkatrészek adatlapjai:

31. *ATmega32*. [online] [cit. 2014-5-11]
<http://www.atmel.com/Images/doc2503.pdf>
32. *MCP9700*. [online] [cit. 2014-5-11]
<https://www.sparkfun.com/datasheets/DevTools/LilyPad/MCP9700.pdf>
33. *LM35*. [online] [cit. 2014-5-11]
<http://www.ti.com/lit/ds/symlink/lm35.pdf>
34. *DS18B20*. [online] [cit. 2014-5-11]
<http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>
35. *HIH5030*. [online] [cit. 2014-5-11]
<http://sensing.honeywell.com/honeywell-sensing-hih5030-5031%20series-product-sheet-009050-2-en.pdf?name=HIH-5030-001>
36. *SHT1x*. [online] [cit. 2014-5-11]
http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf

1. melléklet: program kód

```
#define F_CPU 1000000UL // 1 MHz CPU clock

#include <util/delay.h> // Időzítő, késleltető rutinokat tartalmazó fájl
#include <avr/io.h> // AVR konstansokat, beállításokat tart. fájl
#include <avr/interrupt.h> // Megszakításokat kezelő rutinokat tart. fájl
#include "lcd.h"

volatile uint16_t timer;
volatile uint16_t timex;
volatile uint16_t adc_result0, adc_result1;

// initialize adc
void adc_init(){
    ADMUX = (1<<REFS0); // AREF = AVcc
    ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0); // ADC előosztó 128
}

uint16_t adc_read(uint8_t ch){ // adc érték olvasása függvény
    uint16_t sum=0; // channel kiválasztása 0~7,
    ch &= 0b00000111; // 'ch' AND '7' visszaadja a megfelelő bitet
    ADMUX = (ADMUX & 0xF8)|ch; // alsó 3 bit törlése, majd a 'ch' beírása
    for (int i=0; i<8; i++){
        ADCSRA |= (1<<ADSC); // átalakítás indítása, ADSC = '1'
        while(ADCSRA & (1<<ADSC)); // várunk, míg a konverzió befejeződik, ADSC ismét '0' lesz
        sum+=ADC;
    }
    sum/=8;
    return (sum);
}

ISR(TIMER0_OVF_vect) { // A timer0 túlsordulást kezelő rutin utasításai
    timer++;
    if (timex>0) timex--;
    PORTB ^= (1<<PB2);
    adc_result0 = adc_read(0); // PA0 ADC olvasása
}

ISR(TIMER1_OVF_vect) { // A timer1 túlsordulást kezelő rutin utasításai
    PORTB ^= (1<<PB1);
    adc_result1 = adc_read(1); // PA1 ADC olvasása
}

ISR (INT1_vect) { // INT1 megszakítás kiszolgáló rutin utasításai
    PORTB ^= (1<<PB3);
    if (!(PINC & 0b01000000)) {
        timer=0;
        timex=30;
        GICR &= ~(1<<INT1); //INT1 letiltása (nyomógomb pergés megakadályozása érdekében)
    }
}

void init(){
    cli(); // megszakítások kikapcsolása

    // portok beállítása
    DDRB |= (1<<PB0)|(1<<PB1)|(1<<PB2); // PB0:3 kimenet
    PORTB |= (1<<PB0)|(1<<PB1)|(1<<PB2); // PB0:3 lábra +5V (LED1 kikapcsolva)

    DDRD &= ~(1<<PD3); // PD3 bemenet
    PORTD |= (1<<PD3); // PD3-hoz tartozó felhuzoellenallas be

    DDRC &= ~(1<<PC0)|(1<<PC1)|(1<<PC6); // PC0:1,6 bemenet
    PORTC |= (1<<PC0)|(1<<PC1)|(1<<PC6); // PC0:1,6-hoz tartozó felhuzoellenallas be

    // inicializálja az adc-t es led-t
    adc_init();
    lcd_init(LCD_DISP_ON_CURSOR);

    _delay_ms(50);
}
```

```

//Timer0 and Timer1
TIMSK=(1<<TOIE0) | (1<<TOIE1);           // engedélyezzük mindkét timer túlsordulását
TCNT0=0x00;                               // set timer0 counter initial value to 0
TCCR0 = (1<<CS02) | (1<<CS00);             // a timer0 /1024 előosztóval
TCCR1B |= (1 << CS10) | (1 << CS12);      // a 16 bites timer1 szintén /1024 előosztóval

GICR |= (1<<INT1);                         // az INT1 külső interrupt beállítása
MCUCR |= (1<<ISC11);                       // INT1 engedélyezése (PD2-es láb)
                                           // a láb lefutó élére adjon megszakítást

sei();                                     // megszakítások bekapcsolása
}

int main()
{
    char pinchar;                          //LCD kiírást segítő
    char int_buffer[10];                   //LCD panel segítő

    uint8_t menu_i = 1;                   //menü 1 indításnál
    uint8_t menu_i_old = 10;              //menü old
    int menu_sum;

    int LOW=150;    int HIGH=155;    int KORR=125;
    int VLOW=150;   int VHIGH=155;   int VKORR=125;
    int HLOW=150;   int HHIGH=235;   int HKORR=27;

    menu_sum = 0;
    timer=30;
    init();

    // főprogram
    while(1) {

        if (adc_result0 >= HIGH)    PORTB |= (1<<PB0);
        if (adc_result0 <= LOW)     PORTB &=~(1<<PB0);

        //if (adc_result0 >= VHIGH) PORTB |= (1<<PB1);
        //if (adc_result0 <= VLOW)  PORTB &=~(1<<PB1);

        if (adc_result1 >= HHIGH)    PORTB |= (1<<PB1);
        if (adc_result1 <= HLOW)     PORTB &=~(1<<PB1);

        lcd_clrscr();

        //ha TIMEX menü idő =0 akkor a nem a menüben vagyunk
        if (timex==0) {
            // megszerzett adatok kiírása
            lcd_gotoxy(0,0);
            lcd_puts("Temperature=");

            itoa(adc_result0-KORR, int_buffer, 10);
            lcd_gotoxy(12,0);
            lcd_puts(int_buffer);

            lcd_gotoxy(0,1);
            lcd_puts("RelHumidity=");

            itoa(adc_result1-HKORR, int_buffer, 10);
            lcd_gotoxy(12,1);
            lcd_puts(int_buffer);

            _delay_ms(100);
        }
        //ha TIMEX menü idő >0
        else {
            // menü kiírása i-edik item csillaggal
            lcd_gotoxy(0,0);
            if ( menu_i == 1 )    lcd_puts("TLO* THI  TEMP ");
            if ( menu_i == 2 )    lcd_puts("TLO  THI* TEMP ");
            if ( menu_i == 3 )    lcd_puts("TLO  THI  TEMP*");
            if ( menu_i == 4 )    lcd_puts("HLO* HHI  HUMI ");
            if ( menu_i == 5 )    lcd_puts("HLO  HHI* HUMI ");
            if ( menu_i == 6 )    lcd_puts("HLO  HHI  HUMI*");

            menu_i_old = menu_i;
        }
    }
}

```

```

// menüben + gomb megnyomása i-edik item növelése
if (!(PINC & 0b00000001)) {
    if ( menu_i == 1 )    LOW++;
    if ( menu_i == 2 )    HIGH++;
    if ( menu_i == 3 )    KORR++;
    if ( menu_i == 4 )    HLOW++;
    if ( menu_i == 5 )    HHIGH++;
    if ( menu_i == 6 )    HKORR++;
}

// menüben - gomb megnyomása i-edik item csökkentése
if (!(PINC & 0b00000010)) {
    if ( menu_i == 1 )    LOW--;
    if ( menu_i == 2 )    HIGH--;
    if ( menu_i == 3 )    KORR--;
    if ( menu_i == 4 )    HLOW--;
    if ( menu_i == 5 )    HHIGH--;
    if ( menu_i == 6 )    HKORR--;
}

// menüben változott valami TIMEX=10
if ( LOW + HIGH + KORR + HLOW + HHIGH + HKORR != menu_sum ) {
    timex=30;
    menu_sum = LOW + HIGH + KORR + HLOW + HHIGH + HKORR;
}

// menü item1 értékének kiírása
pinchar = "";
lcd_gotoxy(0,1);
if ( menu_i == 1 || menu_i == 2 || menu_i == 3 )    itoa(LOW-KORR, pinchar, 10);
else    itoa(HLOW-HKORR, pinchar, 10);
lcd_puts(pinchar);

// menü item2 értékének kiírása
pinchar = "";
lcd_gotoxy(5,1);
if ( menu_i == 1 || menu_i == 2 || menu_i == 3 )    itoa(HIGH-KORR, pinchar, 10);
else    itoa(HHIGH-HKORR, pinchar, 10);
lcd_puts(pinchar);

// menü item3 értékének kiírása
pinchar = "";
lcd_gotoxy(10,1);
if ( menu_i == 1 || menu_i == 2 || menu_i == 3 )    itoa(KORR, pinchar, 10);
else    itoa(HKORR, pinchar, 10);
lcd_puts(pinchar);

// menü idő TIMEX értékének kiírása
pinchar = "";
lcd_gotoxy(14,1);
itoa(timex, pinchar, 10);
lcd_puts(pinchar);

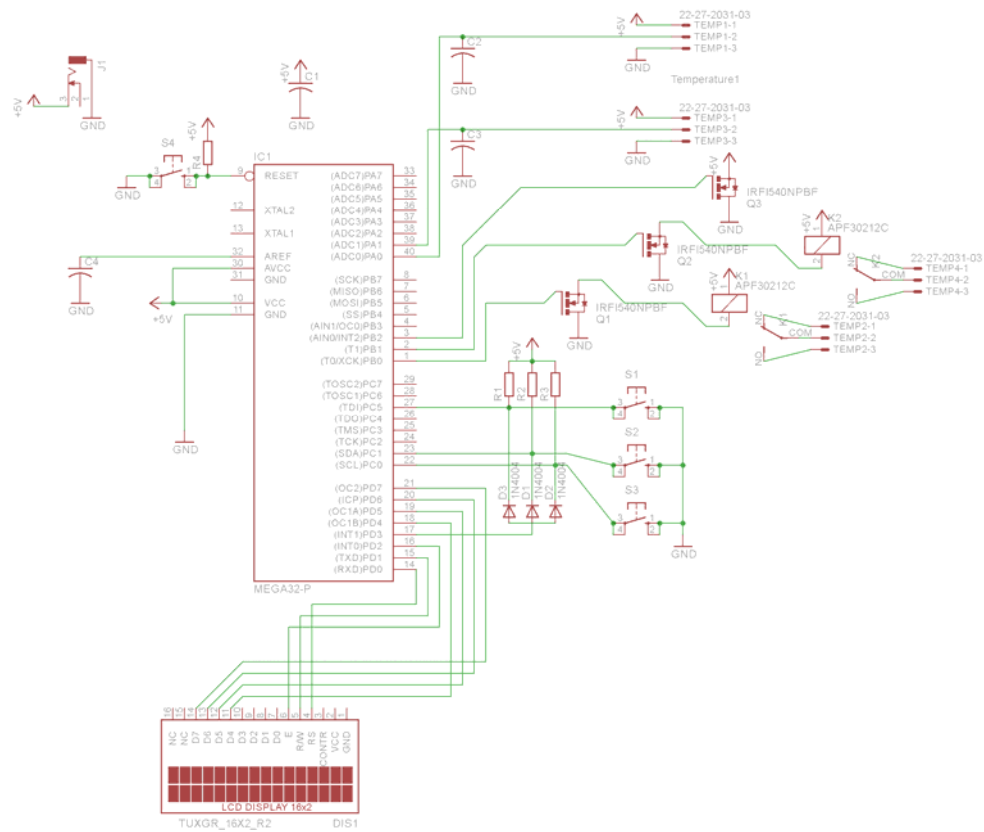
// ha INT0 megszakítás ki van kapcsolva, akkor egy kis várakozás után bekapcsoljuk
if((GICR & (1<<INT1)) == 0) {
    _delay_ms(20);    // várakozás néhány száz ms-ig, amíg a nyomógomb befejezi a pergést

// minden további várakozó megszakítás törlése, majd a megszakítások engedélyezése
GIFR |= (1<<INTF1);    // INT0-hoz tartozó jelzőbit törlése
GICR |= (1<<INT1);    // INT0 megszakítás engedélyezése
}

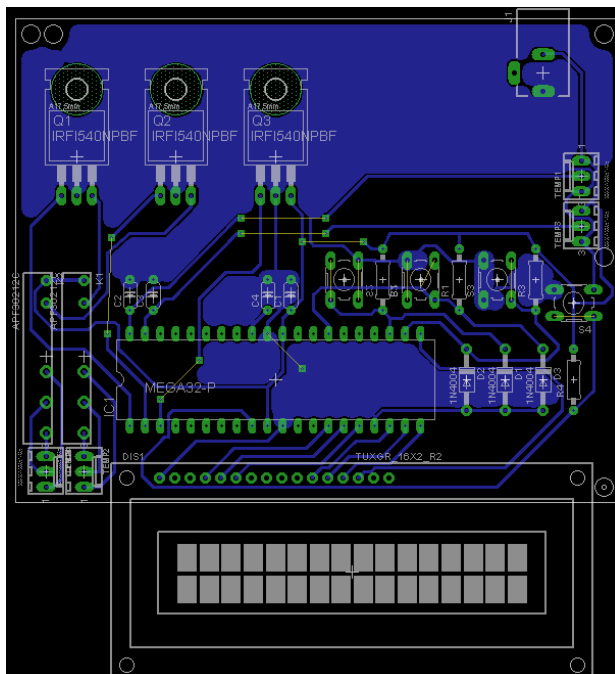
if (!(PINC & 0b01000000)) {
    menu_i++;
    if ( menu_i == 7 )    menu_i = 1;
    timex=30;
}
_delay_ms(25);
}
}
}

```

1.1.melléklet: áramkör bekötési rajza



Bekötési rajz.

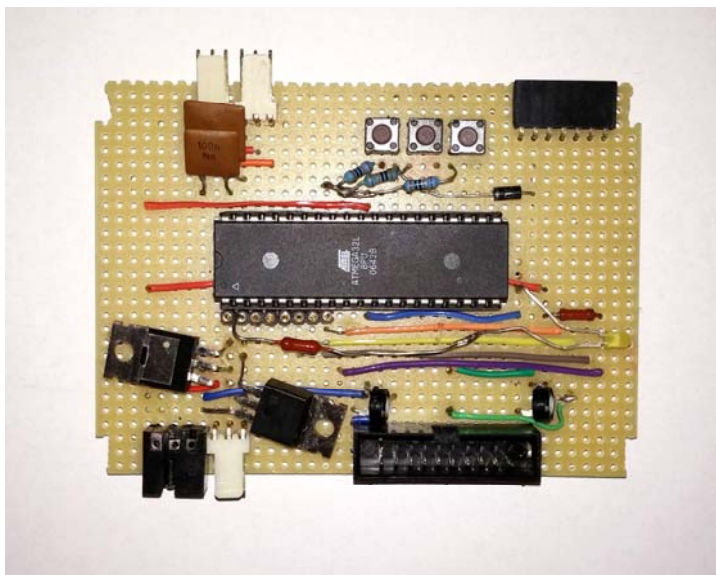


Nyomtatott áramköri lap.

1.2.melléklet: fotódokumentáció



Az elkészült berendezés



Áramköri lap



A doboz belseje és a kijelző



Kijelzés alapeset.



Menü 1



Menü 2



Menü 3



Menü 4



Menü 5



Menü 6