

## Projekt feladat Vincze Dávid ( ITJ3UD )

Napjainkban már rengeteg olyan okos eszköz vesz bennünket körül, amely egyszerűsíti, könnyebbé és gyorsabbá teszi életünket. az utóbbi időben megjelentek olyan olcsó, egyszerűen kezelhető hardver és szoftver eszközök, amikkel mi magunknak is elkészíthetünk ilyen készülékeket. Több ilyen megoldás közül is választhatunk. Én ebben a projektben az Arduino hardver és szoftver eszközökkel megvalósított eszköz segítségével mutatom be, hogy hogyan lehet ezt megvalósítani.

A projekt egy demo "Okos otthon" példáján keresztül mutatja be, hogy hogyan lehet mérési adatokat bekérni szenzorokról, digitális bemenetekről, azokat LCD kijelzőn megjeleníteni és SD kártyán CSV formátumú adatfileban tárolni, ami importálható Excelbe. Továbbá bemutatja hogyan lehet bluetooth kommunikációs csatornán keresztül kezelni a kimeneteket. A rendszer állapota, és a működés közbeni események megjelennek a soros porton is, ahol futásidőben monitorozhatóak. Közben sorra veszem a tervezési szempontokat, és a fejlesztés során tapasztalt problémákat is.

A demo "Okos otthon" az életben előforduló mérési, adatgyűjtési és működtetési feladatokat modellezi.

Hőmérséklet, páratartalom, villamos teljesítmény, készülékek működési állapota, nyílászárók zárt állapota, időkijelzés, készülékek távműködtetése reléekkel okostelefon bluetooth kapcsolaton keresztül.

Olyan méréseket választottam, amelyek menetség roppantul előtérbe kerültek ( hőmérséklet, villamos teljesítmény).

A rendszer állapota kb percenként kiíródik az SD kártyára időbélyeggel, így később részletesen elemezhető Excelbe beimportálva. Ezeken az alapokon aztán már csak a képzelőerőnk szab határt a fejlesztésnek, nem a lehetőségek hiánya.

Azért választottam az Arduino rendszert, mivel könnyen hozzáférhető, nagyon jól dokumentált, és olcsó.

Nagyon sok modul elérhető, és az interneten rengeteg segédanyag található a felhasználáshoz, példák, mintaalkalmazások. Az Arduino rendszer oktatási célokra készült, így egyszerűen lehet vele dolgozni. A keretrendszer ingyenes, és tud magyarul. Egyszerűen kezelhető szoftver, és jól támogatja a hibakeresést. Ezenkívül biztosítja a lefordított program letöltését az alaplapba. Rendelkezik még

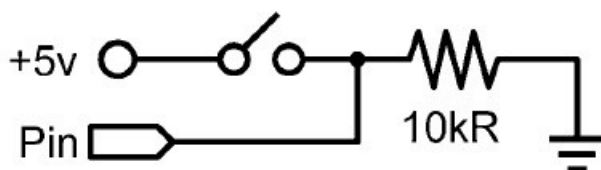
soros monitor funkcióval is ami hasznos segítség a fejlesztéskor, hibakereséskor, valamint a futás figyelemmel kíséréséhez. Ez biztosítja az alkalmazott hardverekhez tartozó alkalmazói könyvtárak karbantartását is. Megnyithatók benne példaprogramok a fejlesztés segítségéhez. minden feladatra lehet többféle megoldást találni, és ezekből kell kiválasztani az adott feladatra leginkább megfelelőt. Különös figyelmet kell szentelni a különböző alkatrészek közötti inkompatibilitásokra. Ez is előfordult a fejlesztés során.

A fejlesztés folyamán először megfogalmaztam a feladatot. A demo "Okos otthon" példáján keresztül bemutatni az Arduino alapvető lehetőségeit a mérés-adatgyűjtés, távműködtetése, adatakijelzés terén.

Igyekeztem egyszerű, könnyen érthető és áttekinthető kódot fejleszteni. Az elkészült projekt használata egyszerű, nem igényel speciális ismereteket. Az alapok megtartásával könnyen továbbfejleszthető.

### **Néhány szó a használt jelekről.**

A készülék a kimeneteken digitális jeleket használ, a bemeneteken pedig digitális és analóg jeleket egyaránt. Esetünkben az analóg jel időben folyamatosan változó feszültség, amely az ACS712 szenzor kimeneti jele. Ezt az A1 analóg bemeneten olvasom be, és ezt adom át egy változónak, amivel már tovább lehet számolni. A digitális bemenetek (kapcsolók) két értéket vehetnek fel 1 és 0. Ezek aktuális értékeit a digitális bemenetek beolvasásával kérem be.



A továbbiakban ezek az értékek már kezelhetők. A digitális kimenetek is két értéket vehetnek fel, 1 és 0, amelyeket a program a kimenetek írásával állít be. Ez működteti a kimenetre kapcsolt relék segítségével a csatlakoztatott készülékeket. Ezáltal tudja működtetni a perifériákat.

## **A projekt megvalósítását a hardver elemek kiválasztásával folytattam.**

- Alaplap Arduino Uno ( később ez Mega2560-ra cseréltem a memóriakapacitás miatt)
- DHT11 hő és páratartalommérő szenzor
- ACS712 árammérő Hall szenzor 5A-es
- LCD kijelző 4 sor 20 oszlop kék-fehér
- I2C illesztő modul az LCD kijelzőhöz
- HC06 bluetooth modul
- Data logger shield SD kártya és RTC
- 2 CSATORNÁS RELÉ MODUL 5V DC
- breadboard, vezetékek, ellenállások, kapcsolók, izzók
- 12V DC tápegység

Az internetről letöltöttem hardverelemek kezelőprogramjait és alkalmazási példáit.

Elkészítettem a program moduljainak terveit. Megterveztem az alaplap csatlakozását a perifériákhoz.

## **A program megvalósításához szükséges modulok :**

### **- könyvtárak és változók definiálása**

### **- setup, program indulási alapbeállítások**

- lcd inicializálás
- soros port inicializálás
- szenzorok inicializálása, kalibrálása
- I2C inicializálás
- Data logger shield ( SD kártya, RTC ) inicializálás
- HC06 bluetooth inicializálás
- digitális kimenetek és bemenetek definiálása és a kimenetek alaphelyzetbe állítása

### **- főciklus**

- idővel és időzítéssel kapcsolatos feladatok
- szenzorok adatainak beolvasása
- digitális bemenetek beolvasása

- HC06 bluetooth olvasása
- digitális kimenetek kezelése
- SD kártya írás
- soros port kezelése
- LCD kiírás

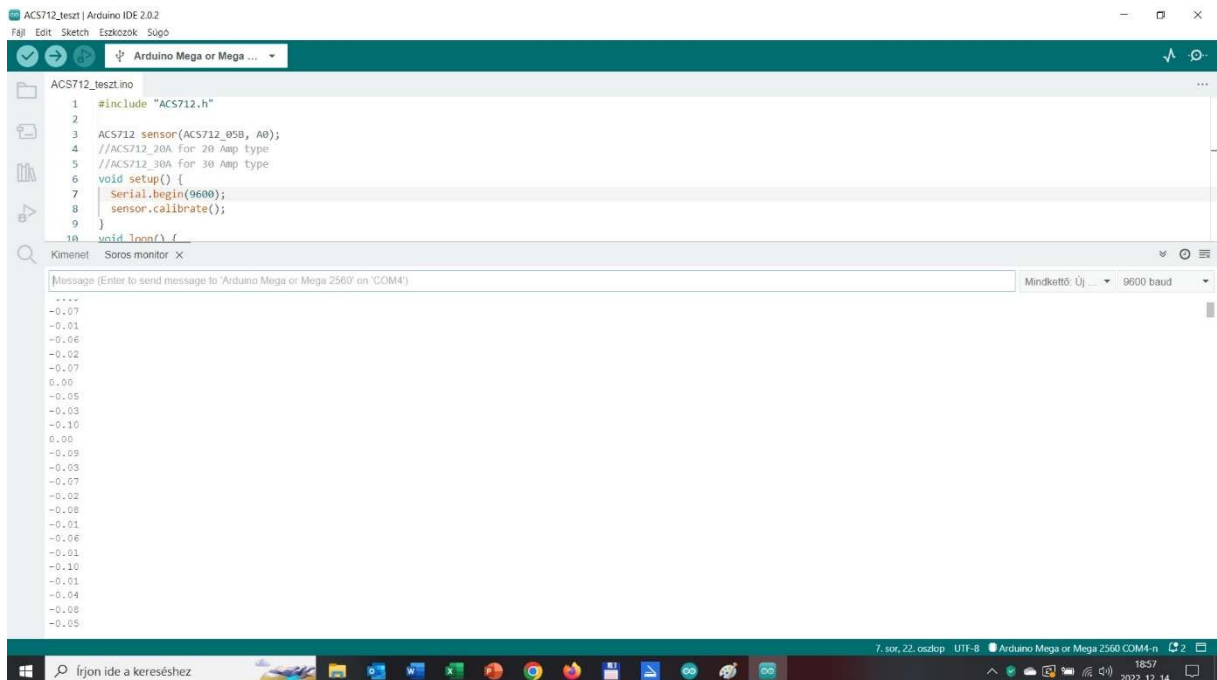
A fejlesztés során felhasználtam az alkatrészek és szoftverek leírásait és az interneten található oktatóanyagokat és leírásokat.

A fejlesztőrendszer Arduino 2.0.2. A felhasznált szoftverkomponensek az internetről szabadon letölthetők.

Az alkatrészek beszerzése után kezdődhetett a projekt megvalósítása.

A megérkezett modulokat egyenként teszteltem. Az ACS712 modul tesztelésekor megállapítottam a nulla korrekciós tényezőt. Ez az áramkör nulla folyó áram esetén tapasztalható pontatlansága.

Esetünkben 0.004A. A program ezt figyelembe veszi a méréskor.

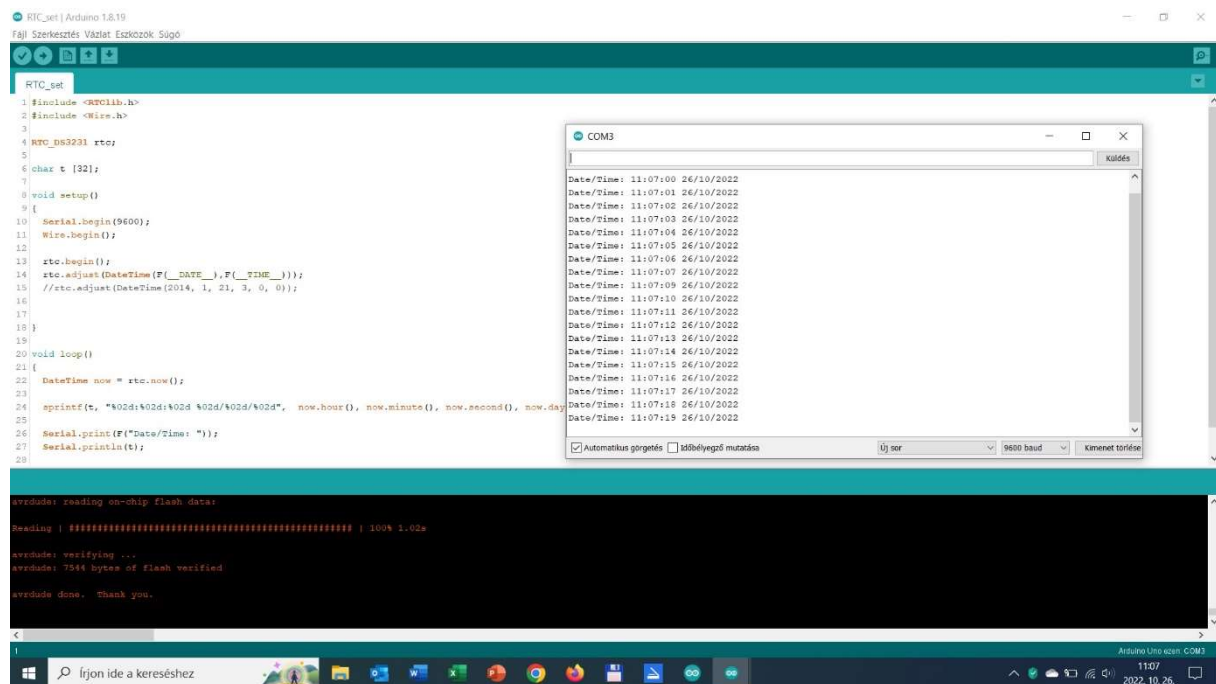


The screenshot displays the Arduino IDE 2.0.2 environment. The main window shows the sketch 'ACS712\_test.ino' with the following code:

```
1 #include "ACS712.h"
2
3 ACS712 sensor(ACS712_05B, A0);
4 //ACS712_20A for 20 Amp type
5 //ACS712_30A for 30 Aamp type
6 void setup() {
7   Serial.begin(9600);
8   sensor.calibrate();
9 }
10
11 void loop() {
```

Below the sketch editor, the 'Kimenet' (Output) window is open, showing the serial monitor. The baud rate is set to 9600. The output displays a series of numerical values, likely representing current measurements, with a mix of positive and negative values (e.g., -0.07, -0.01, -0.06, -0.02, -0.07, 0.00, -0.05, -0.03, -0.10, 0.00, -0.09, -0.03, -0.07, -0.02, -0.08, -0.01, -0.06, -0.01, -0.10, -0.01, -0.04, -0.08, -0.05).

Ugyanígy megtörtént a Data logger shielden található valós idejű óra ( RTC ) beállítása is.



The screenshot shows the Arduino IDE interface. The main window displays the code for `RTC_set.ino`, which includes the `RTClib` and `Wire` libraries, initializes an `RTC_DS3231` module, and sets the time to 2014, 1, 21, 3, 0, 0. The `loop` function prints the current date and time every second. A serial monitor window is open, showing the output of the program, which displays the date and time in the format `Date/Time: DD/MM/YYYY HH:MM:SS`. The output shows the date 26/10/2022 and the time 11:07:00 to 11:07:19. The bottom status bar indicates the board is an `Arduino Uno esp. COM3`.

```
1 #include <RTClib.h>
2 #include <Wire.h>
3
4 RTC_DS3231 rtc;
5
6 char t [32];
7
8 void setup()
9 {
10   Serial.begin(9600);
11   Wire.begin();
12
13   rtc.begin();
14   rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
15   //rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
16
17 }
18
19 void loop()
20 {
21   DateTime now = rtc.now();
22
23   sprintf(t, "%02d:%02d:%02d %02d/%02d/%02d", now.hour(), now.minute(), now.second(), now.day(), now.month(), now.year());
24   Serial.print(F("Date/Time: "));
25   Serial.println(t);
26
27 }
```

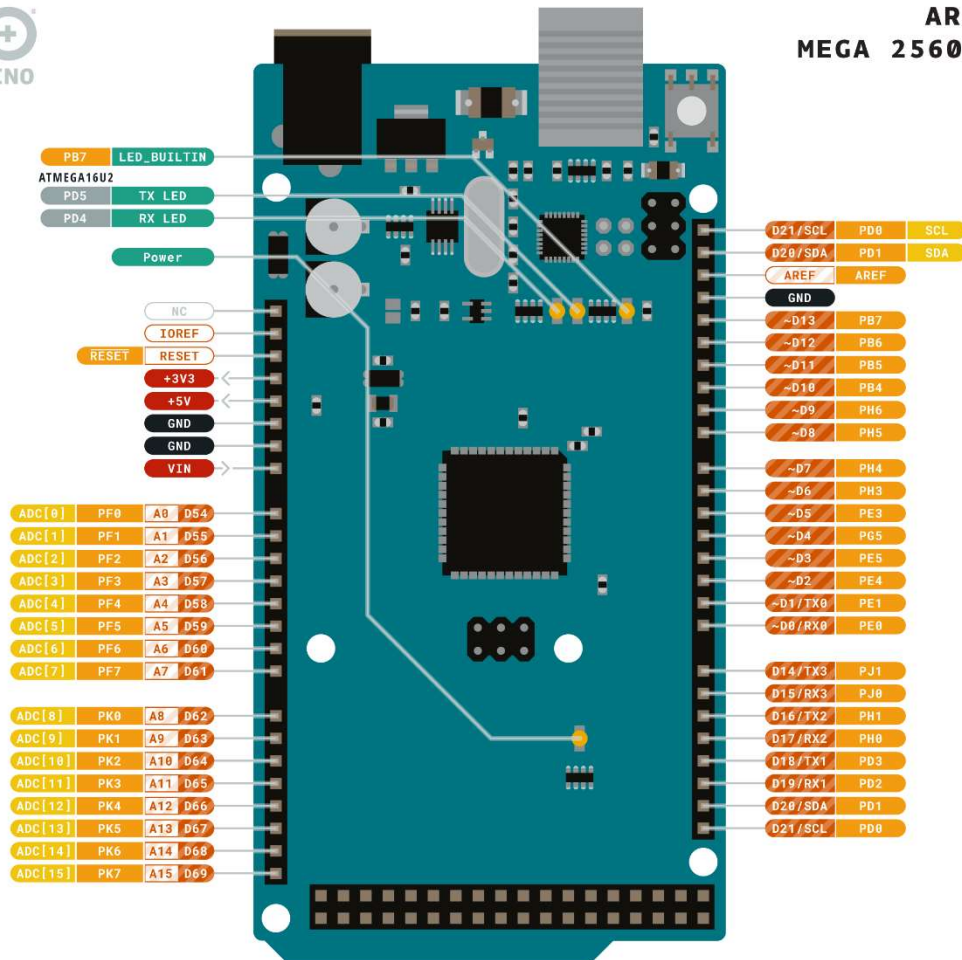
A modulok hibátlanak bizonyultak, így folytattam a berendezés összerakását. Fokozatosan haladtam. Alaplap Data logger shield, SD kártya, LCD, DHT11, ACS712, HC06, Relé modul, bemeneti kapcsolók. A hardver építésével párhuzamosan lépésenként készült a program is. Viszonylag hamar kiderültek olyan dolgok, amik arra indítottak, hogy néhány hardverelemet kicseréljek.

Először az alaplapot cseréltem, mivel vészesen fogyott a memória, és a fordító is figyelmeztetett a várható futási hibákra. Ekkor választottam az Arduino Mega2560-a, mivel ebben már 8kbyte memória van az Uno 2 kbytejával szemben, és az ára sem lényegesen magasabb.

Cserébe sokkal több I/O vonallal rendelkezik. Alkalmazásánál viszont figyelembe kell venni, hogy nem teljesen lábkompatibilis az Unoval. Ennek megfelelően néhány hardver és szoftver komponest cserélni kellett (I2C lábak,SD kezelő könyvtár). Ezek mind megtalálhatók a Data logger shield és az Arduino Mega2560 adatlapján.



## ARDUINO MEGA 2560 REV3



Ground	Internal Pin	Digital Pin	Microcontroller's Port
Power	SWD Pin	Analog Pin	
LED	Other Pin	Default	

ARDUINO.CC



This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

A másik cserélt alkatrész az LCD kijelző volt. Eredetileg 16X2-es LCD kijelzőt használtam, de olyan sok adatot kellett megjeleníteni, hogy váltottam 20X4-es típusra. Ez csak a programban igényelt módosítást.

A hardverrel párhuzamosan a program fejlesztése is folyt. Modulonként tesztelés és javítás, bővítés. Csak akkor léptem tovább, ha a modul már jól működött. Néhány szoftverprobléma elhárítása sok időt és utánjárást igényelt (pl az Uno és a Mega2560 közti különbségek kezelése). Ugyancsak hosszabb időbe telt, míg jól működött az idővel és az időzítésekkel kapcsolatos feladat. Azért van szükség erre hogy előállítsa az adattároláshoz az időbélyeget, az LCD-re kiíráshoz az időt, az adattároláshoz az időzítéshez (1 perc), hogy ne legyen túl nagy az adtbázis méret, és ne legyen túl

nagy az ismétlődés, a kijelző analóg mérések kijelzés sorának ciklikus váltásához, a soros státuszüzenet időzítéséhez ( kb 4 sec ) hogy olvasható legyen. Bizonyos programrészleteket átvettem az internetről, mivel korrekten működnek pl: ékezetes betűk definiálása és kezelése. Ez például laikusok számára is sokkal jobban érthetővé teszi a kezelést.

## **Most nézzük a programkódot.**

### **1. Könyvtárak és változók definiálása**

Ebben a programrészben integráljuk a programba a hardver kezeléséhez szükséges könyvtárakat, és definiálom a működés közben használt változókat, kezdőértéket is adok nekik ha szükséges.

```
#include <dht11.h> // dht11 szenzor könyvtár
```

```
#include "ACS712.h" // Hall szenzor könyvtár
```

```
#include <RTCLib.h> // RTC (valós idejű óra) modul könyvtár
```

```
#include <Wire.h> // WIRE I2C könyvtár
```

```
#include <SPI.h> // SPI kommunikáció könyvtár
```

```
#include <SD.h> // SD kártya könyvtár
```

```
#include <LiquidCrystal_I2C.h> // LCD I2C könyvtár
```

```
RTC_DS3231 rtc; // RTC objektum neve rtc
```

```
#define DHT11PIN 8 //dht11 kimenet az Arduino 8. láb
```

```
dht11 DHT11; //Szenzor objektum neve DHT11
```

```
LiquidCrystal_I2C lcd(0x27,20,4); // I2C LCD beállítás : címe 0x27, 20 oszlop és 4 sor
```

```
ACS712 sensor(ACS712_05B, A1); // árammérő 5A-es típus az A1-es analóg bemeneten
```

```
bool perc_valtas = false; // percváltás detektálása az SD kártya íráshoz
```

```
int lcd_valt=1; // számláló az LCD első második mit írjon ki
```

```
char idobelyeg [20] ; // időbélyeg az adttároláshoz SD kártyán
```

```
char ido [10] ; // idő a kijelzéshez
```

```
int mp ; // másodperc
```

```
int perc ; //perc
```

```
int mp_seged ; // másodperc segédváltozó
```

```
int perc_seged ; // perc segédváltozó
```

```
int mp_szamlalo =0; // változó a másodpercek számolásához
```

```
int soros_szamlalo = 19; // időzítő számláló a soros port írásához
```

```
float I = 0; // áram változó
```

```
const int U = 12; //feszültség konstans 12 V DC
```

```
int homerseklet=0; // hőmérséklet változó
```

```
int paratartalom =0; // páratartalom változó
```



```
float teljesitmeny=0; // teljesítmény változó
```

```
char BT_in_char; //bluetooth bejövő karakter
```

```
bool D_in_1, D_in_2, D_in_3, D_in_4 ; // digitális bemenetek boolean változói
```

```
int D1_pin = 23; // 1. digitális bemenet a D23 láb
```

```
int D2_pin = 25; // 2. digitális bemenet a D25 láb
```

```
int D3_pin = 27; // 3. digitális bemenet a D27 láb
```

```
int D4_pin = 29; // 4. digitális bemenet a D29 láb
```

```
int rele1 = 6; // Relé1 kimenet a D6 láb
```

```
int rele2 = 7; // Relé2 kimenet a D7 láb
```

```
String R1_status = "KI"; // 1. Relé alaphelyzetben kikapcsolva kiíráshoz
```

```
String R2_status = "KI"; // 2. Relé alaphelyzetben kikapcsolva kiíráshoz
```

## 2. Setup

A program alapbeállításainak elvégzése, üdvözlő üzenet.



```
void setup() // beállítások induláskor
{
    Wire.begin(); // I2C inicializálás

    lcd_inicializalas(); // LCD inicializálás, ékezetes betűk definiálása

    lcd.clear(); // LCD törlés
    lcd.setCursor(0,0);
    lcd.print(" Vincze D\010vid "); // induló kijelzés
    lcd.setCursor(0,2);
    lcd.print(" S E T U P "); // induló kijelzés

    Serial.begin(9600); // soros kapcsolat beállítása

    SD_inicializalas(); // SD kártya inicializálás

    RTC_inicializalas(); // RTC inicializálás
    sensor.calibrate(); // ACS712 szenzor kalibrálás

    Serial1.begin(9600); // bluetooth soros port ( serial1 TX1(18), RX1(19)) beállítása
```

```
pinMode(D1_pin, INPUT); // D2 láb bemenet
```

```
pinMode(D2_pin, INPUT); // D3 láb bemenet
```

```
pinMode(D3_pin, INPUT); // D4 láb bemenet
```

```
pinMode(D4_pin, INPUT); // D5 láb bemenet
```

```
pinMode(rele1, OUTPUT); // D6 láb kimenet
```

```
digitalWrite(rele1,LOW); // 1. relé alaphelyzetben kikapcsolva
```

```
pinMode(rele2, OUTPUT); // D7 láb kimenet
```

```
digitalWrite(rele2,LOW); // 2. relé alaphelyzetben kikapcsolva
```

```
delay ( 1000); // 1 mp szünet
```

```
lcd.clear(); // LCD törlés
```

```
}
```

A kódból látható, hogy az egyszerűbb inicializálások itt lettek megírva, de az összetettebbekre saját programrész készült. Ilyen az SD kártya kezelés előkészítése és az LCD inicializálása.

Az SD inicializálása hibakezelést is tartalmaz, pl: nincs, vagy nem írható a kártya. Ekkor a program hibaüzenettel leáll. Az LCD inicializálás során kerülnek definiálásra az ékezetes karakterek. A digitális ki és bemenetek is itt vannak definiálva, és a kimenetek alaphelyzetbe állítva.

### 3. Főciklus

A program futása során folyamatosan ismételt feladatok

```
void loop() // főciklus
{
    ido_feladatok(); // idővel és időzítésekkel kapcsolatos feladatok

    ho_paratartalom_meres(); // hőmérséklet és páratartalom mérés

    teljesitmeny_meres(); // teljesítmény mérés

    digitalis_bemenetek_beolvasasa(); // digitális bemenetek beolvasása

    HC06_bluetooth(); // HC06 bluetooth feladat

    kimenet_kezeles(); // kimenetek kezelése

    if (perc_valtas)
    {
        // hogy ne legyen túl nagy az adat.csv file csak percenként ír bele
        Serial.println("SD írás");
        SD_iras(); // adatsor írás az SD kártyára
        perc_valtas=false;
    }

    soros_szamlalo++;

    if (soros_szamlalo==20) // hogy olvasható legyen a kiírás, kb 4 másodpercenként frissül a rendszer
    állapot
    {
        soros_kiiras(); // adatok kiírása a soros portra
    }
}
```

```

    soros_szamlalo=0; // soros számláló nullázás
}

lcd_kiiras(); // adatok kiírása az LCD kijelzőre

delay(100); // 0,1 másodperc szünet

}

```

Ebben a programkód részben vannak azok az utasítások, amiket a program futása során folyamatosan ismétél.

- elvégzi az idővel és az időzítésekkel kapcsolatos feladatokat
- a szenzorból kiolvassa a hőmérséklet és páratartalom értékét
- az analóg bemenetről beolvasott áramból kiszámolja a villamos teljesítményt
- beolvassa a digitális bemenetek állapotát
- karakter beolvasása a HC06 bluetooth modulból
- a bluetoothon vett karakter alapján elvégzi a kimenetek kezelését
- ha az időfeladatokban percváltást detektál, akkor az aktuális státuszt kiírja a SD kártyára CSV fileba, és a soros vonalon jelzi az írás megtörténtét. Törli a percváltást.
- kb 4 másodpercenként kiküldi a soros portra a státuszjelentés. Azért kell egy számlálóval késleltetni, hogy olvasható legyen.
- a 4 soros LCD-n megjeleníti a státuszt
  1. sor Az RTC-ből kiolvasott idő óó:pp:mm formátumban
  2. sor ciklikusan váltva( kb 5 másodpercenként ) kiírja az analóg méréseket hőmérséklet, páratartalom, villamos teljesítmény
  3. sor a digitális bemenetek állapota
  4. sor a relék állapota

## 4. Inicializálások

### SD kártya inicializálás

```
void SD_inicializalas() // SD kártya inicializálása
{
    if (!SD.begin(10,11,12,13)) // az Arduino Mega2560 miatt meg kell adni a Datalogger shield által
    használt lábakat
    {
        Serial.println(" SD kártya nincs, vagy nem inicializálható"); // ha nincs SD kártya, vagy nem
        inicializálható
        lcd.clear();
        lcd.print(" SD hiba ! ! ! "); /// hiba esetén kilép
        Serial.println(" SD hiba ! ! ! ");
        exit(1); // kilépés
    }
    // ha az SD kártya rendben van, akkor ellenőrzi az adatfile meglétét, ha nincs létrehozza

    if (!SD.exists("adat.csv")) // Ha nem létezik az adat.csv file, akkor létrehozza és beleírja a fejléct
    {
        File adatfile = SD.open("adat.csv", FILE_WRITE);

        adatfile.println("Időbélyeg,Hőmérséklet,Páratartalom,Teljesítmény,Relé1,Relé2,Digit1,Digit2,Digit3,D
        igit4"); //fejléc kiírása a fileba

        adatfile.close(); // adatfile bezárása
    }
}
```

A data logger shielden levő SD kártya használata előtt az Arduino SD könyvtár tartalmát le kellett cserélni olyanra, amiben az SD.begin() függvényben meg lehet adni a használt lábakat. Ez az Arduino Mega2560 miatt kell. Ezután már megfelelően működik.

Ha meg tudja nyitni az SD kártyát, akkor ellenőrzi, hogy van-e rajta adat.csv nevű file.

Ha nincs ilyen file, akkor létrehozza. Ha új a file, akkor beleírja a fejléct és bezárja a file-t.

Ha nem talál SD kártyát, akkor hibaüzenettel kilép a program. A hibaüzenet az LCD-n és a soros porton is megjelenik. " SD hiba ! ! ! "

## LCD inicializálás

```
void lcd_inicializalas() // LCD inicializálás és az ékezetes karakterek definiálása
{
    lcd.init();    // LCD inicializálás
    lcd.backlight(); // háttérvilágítás be
    //kis ékezetes betűk
    byte a1[8] = {B10, B100, B1110, B1, B1111, B10001, B1111};    //á
    byte e1[8] = {B10, B100, B1110, B10001, B11111, B10000, B1110};    //é
    byte i1[8] = {B10, B100, B0, B1110, B100, B100, B1110};    //í
    byte o1[8] = {B100, B100, B0, B1110, B10001, B10001, B1110};    //ó
    byte o2[8] = {B1010, B0, B1110, B10001, B10001, B10001, B1110};    //ö
    byte o3[8] = {B1010, B1010, B0000, B1110, B10001, B10001, B1110};    //ő
    byte u1[8] = {B0010, B0100, B10001, B10001, B10001, B10011, B1101};    //ú
    byte u2[8] = {B1010, B0, B0, B10001, B10001, B10011, B1101};    //ü
    byte u3[8] = {B1010, B1010, B0, B10001, B10001, B10011, B1101};    //ű

    //Ékezetes karakterek definiálása (ö-nek nincs hely!
    {
        lcd.createChar(0, a1); //á
        lcd.createChar(1, e1); //é
        lcd.createChar(2, i1); //í
        lcd.createChar(3, o1); //ó
        lcd.createChar(4, o3); //ő
        lcd.createChar(5, u1); //ú
        lcd.createChar(6, u2); //ü
        lcd.createChar(7, u3); //ű
        lcd.begin(20, 4); //4x20 karakteres LCD
    }
}
```

Az lcd inicializálás használatba veszi az I2C buszon levő 4X20-as kék fehér LCD kijelzőt, bekapcsolja a háttérvilágítást, és definiálja a karakterkészletbe az ékezetes karaktereket. Ezt a kódrészt az internetről vettem át. Csak a kijelző paramétereit módosítottam és jól működik.

### **RTC ( valós idejű óra, real time clock ) inicializálása**

```
void RTC_inicializalas() // RTC valós idejű óra inicializálása
{
    rtc.begin();
}
```

lehetővé teszi a beépített valós idejű óra használatát.

## **5. A főciklusban használt programrészek**

### **Idő feladatok**

```
void ido_feladatok() // idővel és időzítéssel kapcsolatos feladatok
{
    DateTime now = rtc.now();

    sprintf(idobelyeg, "%02d/%02d/%02d %02d:%02d:%02d", now.year(), now.month(), now.day(),
now.hour(), now.minute(), now.second() ); // időbélyeg előállítása ééé/hh/nn ÓÓ:pp:mp
formátumban

    sprintf(ido, "%02d:%02d:%02d" , now.hour(), now.minute(), now.second()); // idő előállítása a
kijelzéshez óó:pp:mp formátumban

    perc = now.minute(); // aktuális perc kiolvasása az RTC-ből
    mp = now.second(); // aktuális másodperc kiolvasása az RTC-ből
    if(perc != perc_seged) // percváltás detektálása
    {
        perc_valtas = true;
        perc_seged = perc;
    }
}
```



```

}
if (mp != (mp_seg)) // 5 másodpercenként váltja az LCD első sorában a kijelzést ciklikusan
    mp_számlalo++;
if ( mp_számlalo == 20 )
{
    mp_számlalo=0;
    {
        lcd_valt++;
        if (lcd_valt > 3)
            lcd_valt = 1;
    }
}
}

```

Ez a kódrész az idővel és az időzítésekkel kapcsolatos feladatokat látja el. Erre azért van szükség, mert az SD kártyán levő adatfile minden adatsora tartalmaz időbélyeget. Ezt a program az RTC-ből kiolvasott információkból állítja elő, formázott éééé/hh/nn óó:pp:mp alakban. Az LCD kijelzőn megjelenő időt is itt állítja elő, ugyancsak formázott óó:pp:mp formátumban.

Ezután a program futása során használt időzítések kezelése következik. Az SD kártyára írás időzítése a percváltás figyelésével működik. Ez úgy működik, hogy egy korábban eltárolt segédváltozóban levő értéket összehasonlít az aktuálisan RTC-ből kiolvasott adattal, és ha különböznek, akkor igazra állítja a változót. Ezt észlelve a program a főciklusban elvégzi az adatsor kiírását az SD kártyán levő .csv fileba. Ezután a változót újra hamisra állítja. A kiírás kezdete és vége kiíródik a soros portra is az időbélyeggel együtt. Hasonló elven működik az LCD kijelző második sorában az analóg értékek ciklikus váltása is. Itt egy változó értéke inkrementálódik, és ettől függően változik a kiírt paraméter.

### **Hőmérséklet és páratartalom mérés**

```

void ho_paratartalom_meres() // hőfok és páratartalom mérés
{
    int chk = DHT11.read(DHT11PIN); // adatok beolvasása a DHT11 szenzorból

```

```
paratartalom = (DHT11.humidity); // Páratartalom kiolvasása a DHT11 szenzorból
```

```
homerseklet = (DHT11.temperature); // Hőmérséklet kiolvasása a DHT11 szenzorból
```

```
}
```

Ebben a programrészben a hőmérséklet és a páratartalom értékek kiolvasása történik a DHT11 szenzorból, és eltárolása a változóknak.

### **Teljesítmény mérés**

```
void teljesitmeny_meres() //Teljesítménymérés
```

```
{
```

```
    I = sensor.getCurrentDC(); // egyenfeszültség (DC) áram beolvasása a Hall szenzorból
```

```
// I = sensor.getCurrentAC(); // váltófeszültség (AC) áram beolvasása a Hall szenzorból
```

```
    if (I < 0.004) // figyelmen kívül hagyja, ha az áram 0.004A-nál kevesebb (teszteléssel megállapított érték)
```

```
    {
```

```
        I = 0;
```

```
    }
```

```
    teljesitmeny = (U*I); // teljesítmény számolása 12V DC feszültséggel számolva
```

```
}
```

Itt történik az ACS712 Hall generátoros árammérő szenzor kimenőjelének beolvasása az A0 analóg bemenetről.

Ebből az áramjelből és a konstansként definiált U (12) V DC jelből számolom ki a teljesítményt. A mérésakor figyelembe veszem a teszt során megállapított mérési pontatlanságot. ( 0.008A ) A panelen a méréshez 12V egyenfeszültséget ( DC ) használok az érintésvédelem érdekében. Így kizárható a véletlen baleset.

### **Digitális bemenetek beolvasása**

```
void digitalis_bemenetek_beolvasasa() // digitális bemenetek beolvasása
```

```
{
```

```
    D_in_1 = digitalRead(D1_pin); // 1. digitális bemenet beolvasása
```

```

D_in_2 = digitalRead(D2_pin); // 2. digitális bemenet beolvasása
D_in_3 = digitalRead(D3_pin); // 3. digitális bemenet beolvasása
D_in_4 = digitalRead(D4_pin); // 4. digitális bemenet beolvasása
}

```

Egyszerű beolvasásokkal beolvasom a digitális bemenetek aktuális értékeit.

### Soros kiíratás

```

void soros_kiiras() // rendszerállapot kiírása a soros portra
{
    Serial.println("----- Rendszer állapot -----");
    Serial.println(idobelyeg);
    Serial.print("Paratartalom (%): ");
    Serial.println(paratartalom); // Páratartalom kiírása a soros vonalra
    Serial.print("Homerseklet (C): ");
    Serial.println(homerseklet); // Hőmérséklet kiírása a soros vonalra
    Serial.print("Teljesítmény : ");
    Serial.print((teljesitmeny));
    Serial.println(" W"); // teljesítmény kiírása a soros vonalra
    Serial.print("Rele1 : ");
    Serial.println(R1_status);
    Serial.print("Rele2 : ");
    Serial.println(R2_status);
    Serial.print("Digitális bemenetek : ");
    Serial.print(D_in_1);
    Serial.print(D_in_2);
    Serial.print(D_in_3);
    Serial.println(D_in_4 );
}

```

A főciklusban időzítve, kb 4 másodpercenként a soros portra kiírom az aktuális rendszerállapotot.

Az időzítésre az olvashatóság miatt van szükség.

### **Bluetooth feladat**

void HC06\_bluetooth() // bluetooth feladat. Serial1-ről karakter beolvasás

```
{  
    BT_in_char = Serial1.read(); // egy karakter beolvasása a HC06 bluetooth modulból  
}
```

Ebben a programrészben olvasok be a HC06 bluetooth modulból egy karaktert. A modul az Arduino Mega 2560 Serial1 portjára csatlakozik. TX1(18) RX1(19).

### **Digitális kimenetek kezelése**

void kimenet\_kezeles() // ha van értelmezhető karakter, akkor kimenet kezelés és az aktuális idő és állapot kiírása a soros portra

```
{  
    //relé1 kezelése  
    if(BT_in_char == 'a') // ha a karakter "a" rele1 bekapcsol  
    {  
        digitalWrite(rele1,HIGH);  
        R1_status = "BE";  
        Serial.print(" Időbélyeg ");  
        Serial.println(idobelyeg);  
        Serial.print("Bluetooth vett karakter : ");  
        Serial.println(BT_in_char);  
        Serial.print("Relé1 : ");  
        Serial.println(R1_status);  
    }  
    if(BT_in_char == 'b') // ha a karakter "b" rele1 kikapcsol  
    {  
        digitalWrite(rele1,LOW);  
        R1_status = "KI";  
        Serial.print(" Időbélyeg ");  
        Serial.println(idobelyeg);  
    }  
}
```

```

Serial.print("Bluetooth vett karakter : ");
Serial.println(BT_in_char);
Serial.print("Relé1 : ");
Serial.println(R1_status);
}
//relé2 kezelése
if(BT_in_char == 'c') // ha a karakter "c" rele2 bekapcsol
{
    digitalWrite(rele2,HIGH);
    R2_status = "BE";
    Serial.print(" Időbélyeg ");
    Serial.println(idobelyeg);
    Serial.print("Bluetooth vett karakter : ");
    Serial.println(BT_in_char);
    Serial.print("Relé2 : ");
    Serial.println(R2_status);
}
if(BT_in_char == 'd') // ha a karakter "d" rele2 kikapcsol
{
    digitalWrite(rele2,LOW);
    R2_status = "KI";
    Serial.print(" Időbélyeg ");
    Serial.println(idobelyeg);
    Serial.print("Bluetooth vett karakter : ");
    Serial.println(BT_in_char);
    Serial.print("Relé2 : ");
    Serial.println(R2_status);
}
delay(10);
}

```

A digitális kimenetek kezelése az induláskori beállítások után attól függ, hogy a bluetooth vevőből milyen karakter érkezik.

Ha a akkor az 1-es relé bekapcsol

Ha b akkor az 1-es relé kikapcsol

Ha c akkor a 2-es relé bekapcsol

Ha d akkor a 2-es relé kikapcsol

A működtetés a megfelelő digitális kimenetek írásával megy végbe.

Ha értékelhető karaktert olvas be a bluetooth modulból és kimenetkezelés történik, akkor ennek ténye a soros porton soron kívül időbélyeggel megjelenik.

### **SD kártya írás**

```
void SD_iras() // SD kártya írása soronként
{
    // adatfile megnyitása
    Serial.println (" SD kártya írás kezdet"); // kiírás a soros portra

    File adatfile = SD.open("adat.csv", FILE_WRITE);

    if (adatfile)
    {
        adatfile.print(idobelyeg); //időbélyeg írása az SD kártyára
        adatfile.print(","); //mezőelválasztó "," írása
        Serial.print(" Időbélyeg "); // időbélyeg kiírása a soros portra
        Serial.println(idobelyeg);
        adatfile.print(homerseklet); //hőmérséklet írása az SD kártyára
        adatfile.print(","); //mezőelválasztó "," írása
        adatfile.print(paratartalom); //páratartalom írása az SD kártyára
        adatfile.print(","); //mezőelválasztó "," írása
        adatfile.print(teljesitmeny); //teljesítmény írása az SD kártyára
```

```

adatfile.print(","); //mezőelválasztó "," írás
adatfile.print(R1_status); //Relé1 állapot írása az SD kártyára
adatfile.print(","); //mezőelválasztó "," írás
adatfile.print(R2_status); //Relé2 állapot írása az SD kártyára
adatfile.print(","); //mezőelválasztó "," írás
adatfile.print(D_in_1); // 1. digitális bemenet állapot írása az SD kártyára
adatfile.print(","); //mezőelválasztó "," írás
adatfile.print(D_in_2); // 2. digitális bemenet állapot írása az SD kártyára
adatfile.print(","); //mezőelválasztó "," írás
adatfile.print(D_in_3); // 3. digitális bemenet állapot írása az SD kártyára
adatfile.print(","); //mezőelválasztó "," írás
adatfile.println(D_in_4); // 4. digitális bemenet állapot írása az SD kártyára
adatfile.close(); //adatfile lezárása
Serial.println(" SD kártya írás vége"); // kiírás a soros portra
delay(50); // 50 ms szünet
}
else
{
    Serial.println(" Adat.csv nem írható"); // ha az adat.csv kártya nem írható
    lcd.clear();
    lcd.print(" SD hiba ! ! ! "); // ha nincs meg az adatfile
    Serial.println(" SD hiba ! ! ! ");
    exit(1); // kilépés
}
}

```

Ha a főciklus azt érzékeli, hogy a percváltás változó igaz, akkor az aktuális állapotot időbélyeggel rögzíti az SD kártyán levő adat.csv fileba. Ez a következőképp történik.

- megnyitja írásra az adat.csv feilet.
- sorban beleírja az időbélyeget, a hőmérséklete, a páratartalmat, a villamos teljesítményt, a kimenetek állapotát, a digitális bemenetek állapotát.

A mezők közé "," elválasztó jel kerül. Így hozza létre adatsorokat ( rekord ).

- az írás kezdetét és végét időbélyeggel együtt soron kívül kiírja a soros portra is.

- ha nem sikerül az írás, hibaüzenettel kilép.

a hibaüzenet megjelenik a soros porton és az LCD kijelzőn is . " SD hiba ! ! ! !"

### **LCD kiírás**

```
void lcd_kiiras() // kiírás az LCD kijelzőre
```

```
{
```

```
// LCD első sorának kezelése
```

```
  lcd.setCursor(0,0); // LDC kurzor 0. sor 1. oszlop
```

```
  lcd.print("Id\004 : "); // RTC idő kiírása
```

```
  lcd.print(ido);
```

```
  lcd.print("      ");
```

```
// LCD harmadik sorának kezelése, digitális bemenetek
```

```
  lcd.setCursor(0, 2);
```

```
  lcd.print("D-be 1:"); // digitális bemenetek állapotának kiírása
```

```
  lcd.print(D_in_1); // 1.digitális bemenet állapotának kiírása
```

```
  lcd.print(" 2:");
```

```
  lcd.print(D_in_2); // 2.digitális bemenet állapotának kiírása
```

```
  lcd.print(" 3:");
```

```
  lcd.print(D_in_3); // 3.digitális bemenet állapotának kiírása
```

```
  lcd.print(" 4:");
```

```
  lcd.print(D_in_4); // 4.digitális bemenet állapotának kiírása
```

```
// LCD negyedik sorának kezelése, reléállapotok
```

```
  lcd.setCursor(0, 3);
```

```
  lcd.print("Rel\0011 ");
```

```
  lcd.print(R1_status); // R1 relé állapot
```

```
  lcd.setCursor(10, 3);
```

```
  lcd.print("Rel\0012 ");
```



```

lcd.print(R2_status); // R2 relé állapot

switch (lcd_valt) // a második sor tartalma az lcd_valt értékétől függ, analóg mérések
{
    case 1: // hőmérséklet

        lcd.setCursor(0,1); // LDC kurzor 2. sor 1. oszlop

        lcd.print ("H\004m\001rs:\001klet:");

        lcd.print(homerseklet);

        lcd.print(" C ");

        break;

    case 2: // páratartalom

        lcd.setCursor(0,1); // LDC kurzor 2. sor 1. oszlop

        lcd.print ("P\010ratartalom:");

        lcd.print(paratartalom);

        lcd.print(" % ");

        break;

    case 3: // teljesítmény

        lcd.setCursor(0,1); // LDC kurzor 2. sor 1. oszlop

        lcd.print("Teljes\002tm\001ny:");

        lcd.print(teljesitmeny);

        lcd.print("W");

        break;

}
}

```

A program futása közben keletkező mérési eredmények és információk az LCD kijelzőn jelennek meg.

Az első sorban az RTC-ből kiolvasott idő.

A második sorban az analóg mért és számított értékek cikikusan ismétlődve, kb 5 másodprcenként.

Az idő feladatokban beállított változó ( lcd\_valt ) értéke határozza meg, hogy mikor melyik.

A harmadik sorban a digitális bemenetek aktuális állapota látszik.

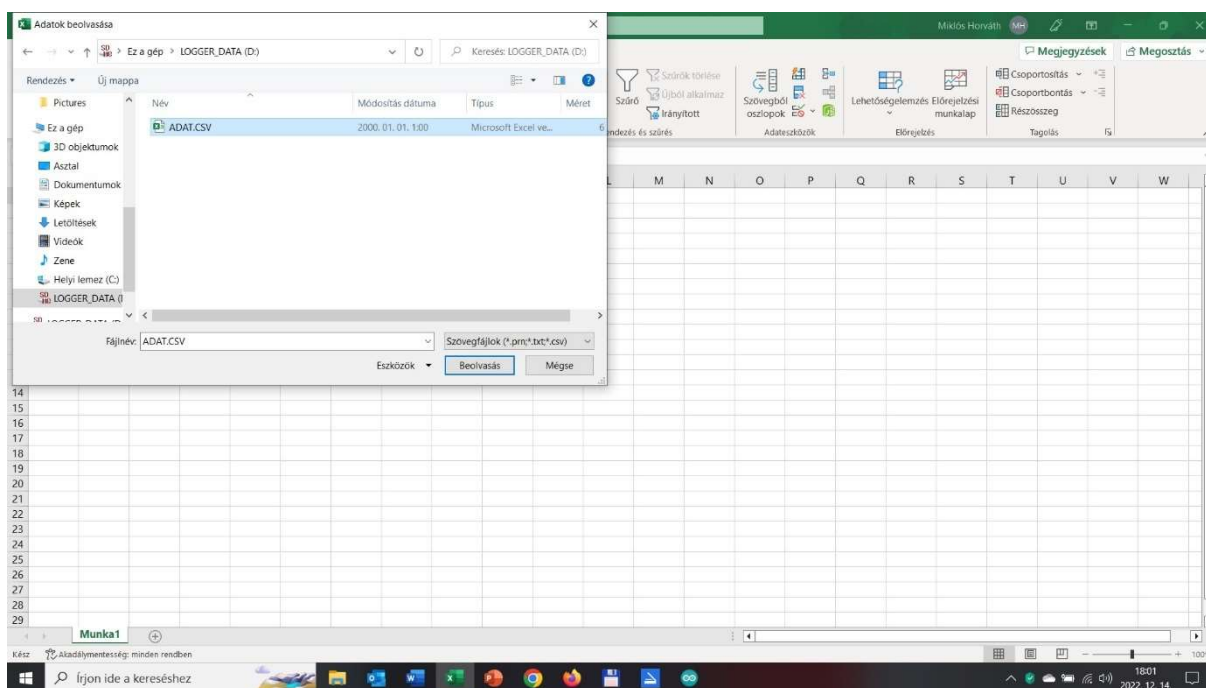
A negyedik sorban a relék aktuális állapota van kijelevve.



Az LCD kijelző teremt kapcsolatot a berendezés és az ember között. Itt jelennek meg az információk.

### A csv file importálása Excelbe

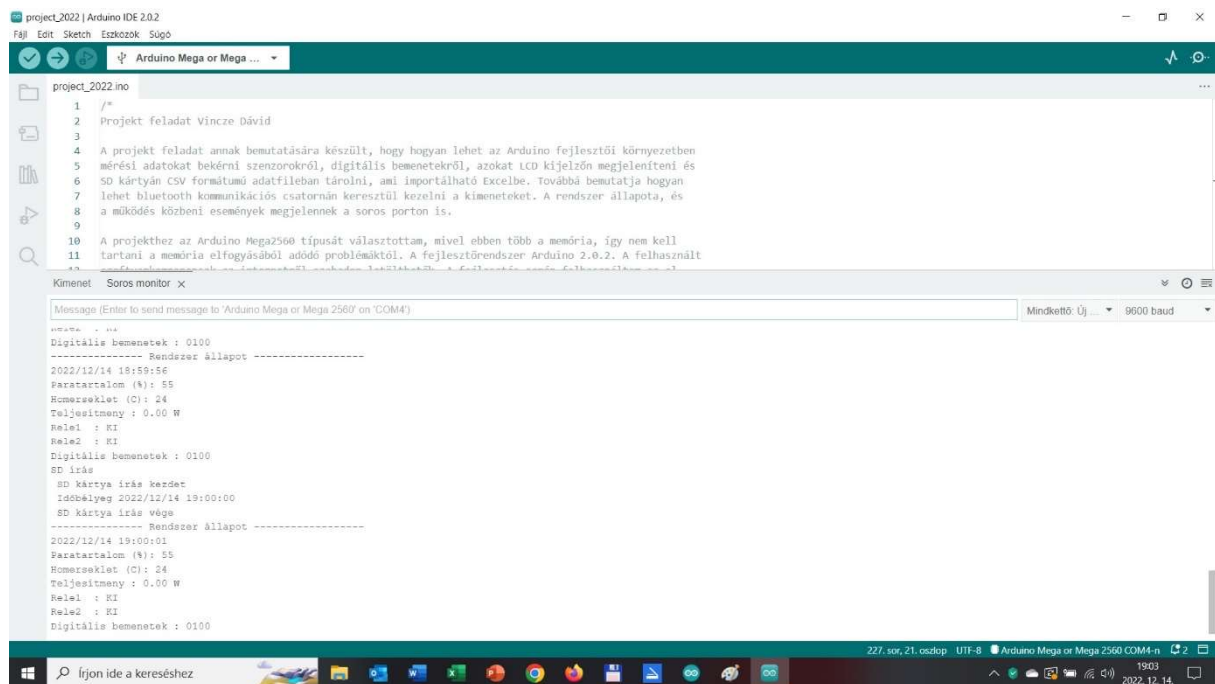
Miután a program elkészült, a készülék folyamatosan tölti fel adatokkal az adat.csv filet. A kikapcsolás után az SD kártyát ki lehet venni a data logger shieldből. Az excel elindítása után be kell tallózni a file-t. ( Adatok-> Szövegből vagy CSV fileből )



Ezután megjelenik az előnézeti kép.

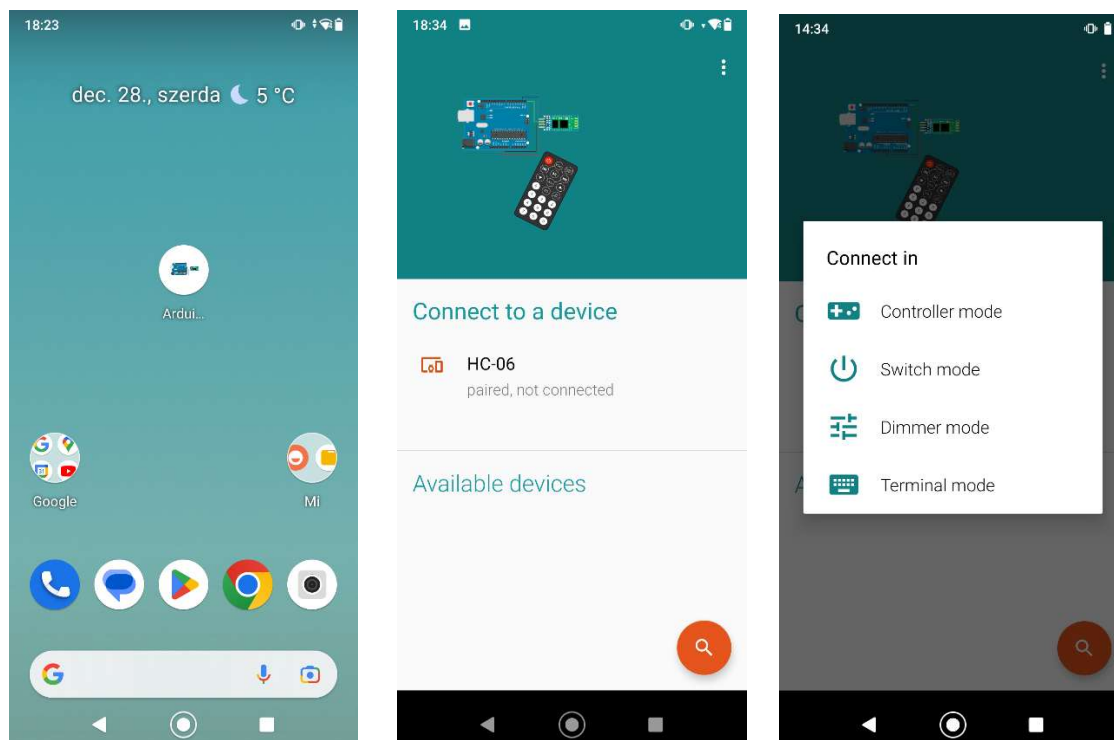


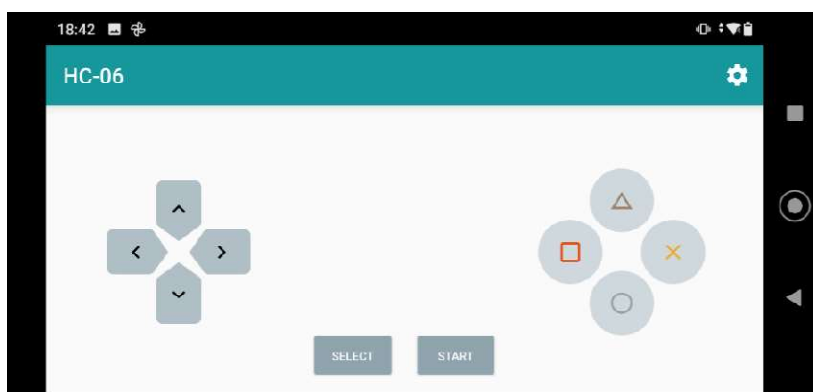
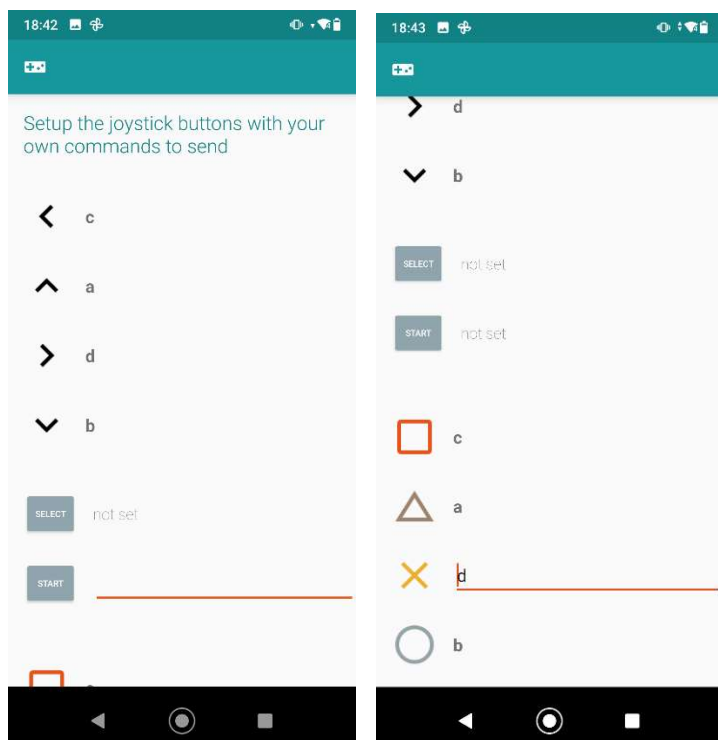
A program futása során a soros portra folyamatosan kb 4 másodpercenként kiírja a státuszt. Itt jelennek meg a reléműködtetés és az SD kártya írás naplózásai is. Ezek megjelenítése független az időzítéstől.



## Android

A bluetooth távirányító feladatát androidos mobiltelefonnal oldottam meg. A Play áruházból letöltött Arduino bluetooth controller appal. A HC06 párosítása után Controller módban definiáltam a billentyűkhöz rendelt karaktereket.



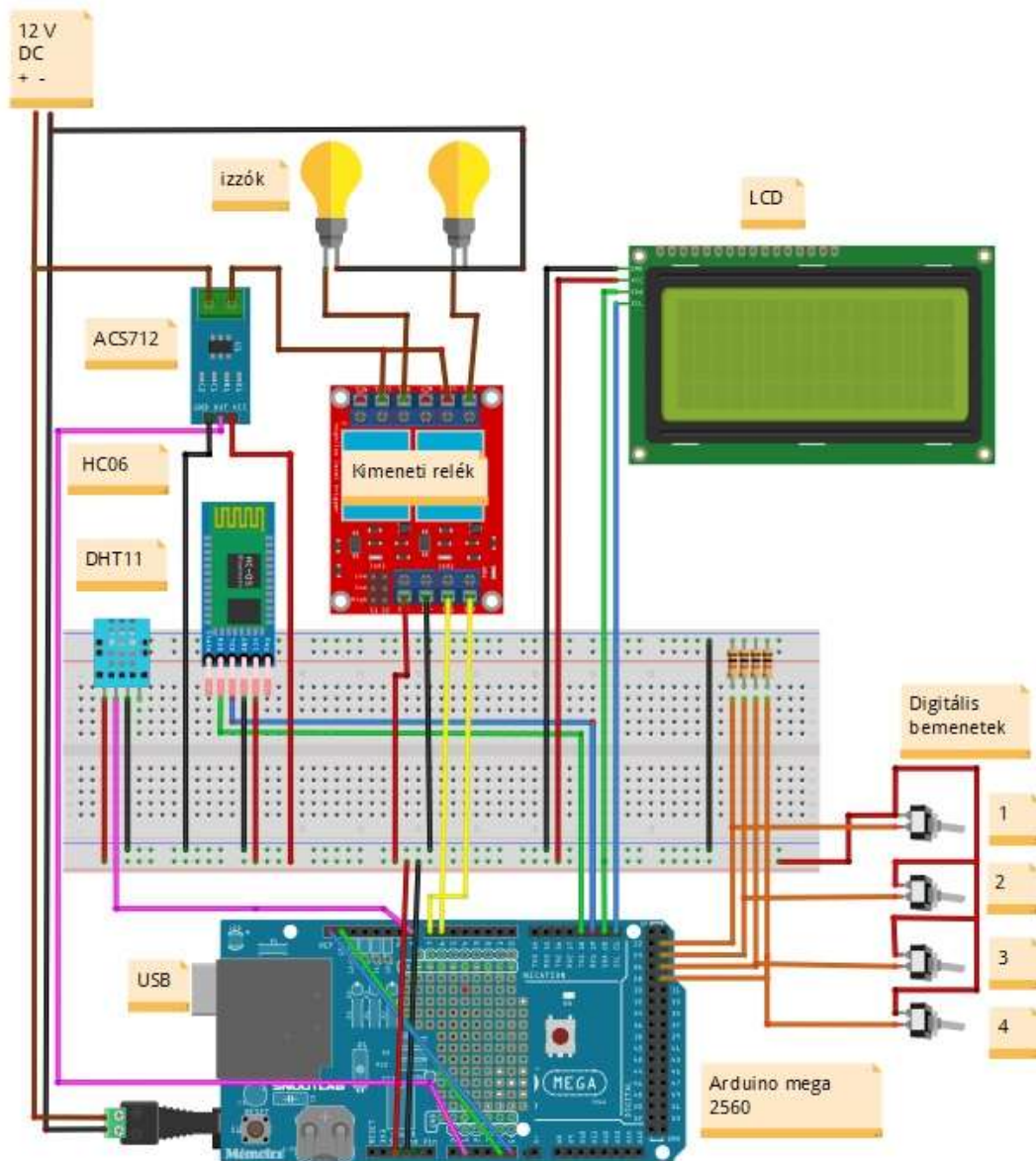


Ezután már működik a távvezérlés.

A berendezés elkészítése során a legtöbb gondot a csatlakozók érintkezési problémái jelentették. Miután ez kiderült, már egyszerűbb lett a hibakeresés és javítás.

### Az áramkör

Az áramkör tervét a Fritzing áramkörtevező szoftver 0.9.3. free beta verziójával készítettem, az interneten talált használati utasítások alapján. A szükséges alkatrészeket a Fritzing weboldaláról töltöttem le.



fritzing