# Pyrheliométer Nap-követő rendszerrel

***SunFlower***

*Bejó Mátyás, Hargitai Benke, Sztojka Áron*
*Felkészítő tanár: Lang Ágota*
*Széchenyi István Gimnázium, Sopron*

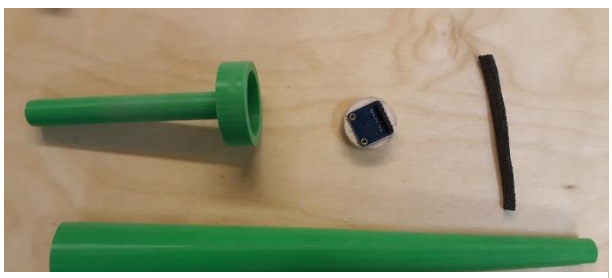
## 1. Bevezetés

A Magyar Asztronautikai Társaság ebben az évben „Csillagunk, a Nap" címmel hirdette meg diákpályázatát. Csapatunk 2/3 része ezen is indult, és azt a kérdést járta körbe, hogy mennyire függünk itt a Földön a Naptól. Ennek során találkoztunk a napállandó fogalmával, így vetődött fel az ötlet, hogyan lehetne ezt az értéket megmérni. Utánanéztünk, és egy pyrheliométer nevű eszközt találtunk erre a célra. Utána olvasva, úgy döntöttünk, hogy készítünk egy saját példányt, hozzá pedig egy Nap-követő rendszert. A pyrheliométer gyakorlatilag egy fél méteres cső, ennek a végében helyezkedik el egy szenzor. Mivel a Nap irányából érkező direkt sugárzást méri, ezért a szenzor felületére merőlegesen kell beesnie a napsugaraknak, emiatt szükséges a Nap követése.

## 2. Eszközeink felépítése

### 2.1 Pyrheliométer

A vezérlést egy Arduino Mega végzi, mivel sok pinre van szükségünk egyrészt a szenzorokhoz – ebből hármat használunk -, illetve a két motor valamint az elektronika számára. A hőmérséklet mérésére egy TMP006 típusú, érintkezést nem igénylő infra termopile szenzort alkalmazunk. (Ez egy olyan eszköz, ami a hőt elektromos energiává és így elektromos jellé alakítja.) A szenzorral egy fekete felületű kis tömegű alumíniumlemez hőmérsékletét mérjük, amely a nap sugárzásának van kitéve. Hogy az égfelület többi részéről



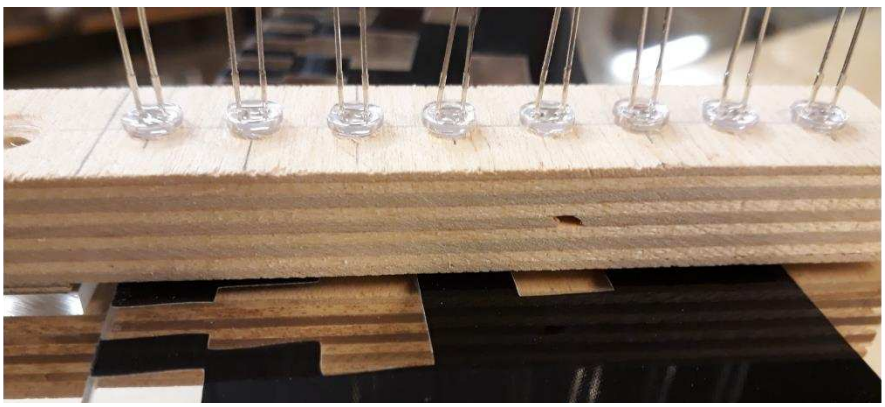
**1. ábra: A pyrheliométer háza még 2 darabban és a termopile szenzor**

érkező szórt fényt kiszűrjük, egy 5° látószögű 3D nyomtatott árnyékoló végében helyeztük el. (1. ábra) Ebből következően fontos az eszköz pozícionálása, ±0,5° pontossággal kell ráállítani a napra. A lemez hőmérsékletének a változásából következtethetünk a beeső teljesítményre tehát a napállandóra.

### 2.2 Nap-követő rendszerünk

Ehhez a Nap pozíciójára állást használjuk (fényerősség-mérés helyett). A szükséges képletet ezen a honlapon találtuk meg: https://www.pveducation.org/pvcdrom/properties-of-sunlight/the-suns-position Ezek az összefüggések a dátummal, helyi idővel és a földrajzi koordinátákkal számolnak, tehát ezeket az adatokat kell tudnunk. Ennek lekérdezésére egy Adafruit GPS modult alkalmaztunk.

Ha megvannak a Nap égi koordinátái (azimuth és magasság (altitude/elevation)), már „csak" be kell lőni rá a pyrheliométert. Az azimutra állás iránytű-szenzor segítségével történik: egy léptetőmotor elforgatja kerekein a kis asztalt a teljes eszközzel, míg a másik az elevation tengelyen már csak a műszert. A magasság szögének a beállításához egy 8-bites encodert használunk. A kód egy negyedkör alakú plexilemezen helyezkedik el, a „mintát" AutoCAD-ben terveztük meg. A minta fekete csíkjai jelentik az 1-es számot, a fehér/átlátszó részek pedig a 0-át jelölik. A hagyományos egyesével növekvő beosztással ellenétben olyan módszert alkalmaztunk, hogy szomszédos számok csak 1 bitben térjenek el egymástól, ezt hivatalosan Gray kódnak nevezik. A lemez egyik oldalán 8 infra-LED, másik oldalán velük szemben 8 fototranzisztor helyezkedik el. (2. ábra) Ezek „olvassák" a kódot. Ez azt jelenti,



**2. ábra: Az encoder a szerelőasztalon, egyik oldalán a LED-ek, másikon a fototranzisztorok, még beforrasztás előtt**

hogy a 90 fokot 28=256 részre osztottuk, ilyen pontossággal tudjuk a magasság szögét beállítani egy másik léptetőmotorral, amelynek az encoder mondja meg, mennyit kell elfordulnia.

## 3. A program

Első lépésként a program lekérdezi a GPS-modultól a szükséges adatokat. A GPS-el való kommunikációhoz saját arduino könyvtárat fejlesztettünk, amellyel $GPRMC típusú mondatokat dekódolunk.

Ezután kiszámoljuk a Nap helyzetét a fentebb említett honlapon található egyenletekkel a földrajzi pozíció és az aktuális dátum és idő alapján. Először a pontos helyi napidőt határozzuk meg, amelyhez egy listában eltároljuk az egyes hónapok napszámát. Majd különböző korrekciós értékek segítségével a Nap égi koordinátáit, vagyis az azimut és magasság szögeit számítjuk ki. (3. ábra)

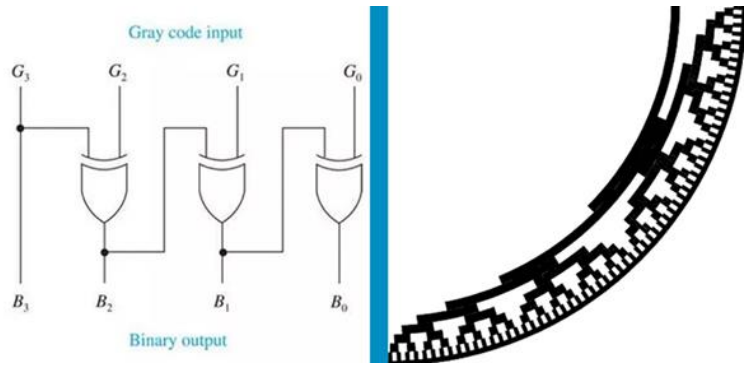
```
void Calc(){
  if(Year%4 == 0){
    for(int i = 0; i > Month-1; i++) d+= ly[i];
    d+=Day;
   } else{
    for(int i = 0; i > Month-1; i++) d+= nly[i];
    d+=Day;
   }

  dTgmt = Long/15;
  b = 360/365*(d-81);
  EoT = 9,87*sind(2*b)-7,53*cosd(b)-1,5*sind(b);
  LT = Hour+Min/60+Sec/3600+dTgmt;
  LST = LT + EoT/60;
  HRA = 15*(LST-12);
  ang =  23,45*sind(360/365*(d-81));
  Elevation = asind(sind(ang)*sind(Lat)+cosd(ang)*cosd(HRA));
  Azimuth = acosd((sind(ang)*cosd(Lat)-cosd(ang)*cosd(HRA))/cosd(Elevation));
}
```

**3. ábra: Kódrészlet a számításokból**

Ezután a program átadja ezeket az értékeket a motorok számára. Az első steppermotor addig forgatja a berendezést, amíg az iránytűszenzor alapján rááll a megfelelő azimutra.

A másik steppermotor számára a Gray-kód ad információt arról, hogy mekkora magassági szögnél jár a műszer. A Gray-kód olvasásához először a szomszédos beosztásban lévő skálát vissza kell alakítani egyenletesen növekvő kettes számrendszerbeli beosztásra. Ezt a feladatot egy olyan logikai kombinációs hálózat látja el, mint ami az ábrán is látható - annyi különbséggel,

hogy a miénk 8 bitet is tud kezelni. Ezt az átalakítást a program 7db XOR logikai művelettel hajtja végre. (4. ábra)




**4. ábra: A logikai műveletek illetve az encoderen található minta**

```
int getPos(int pin7, int pin6, int pin5, int pin4, int pin3, int pin2, int pin1, int pin0) {  //MSB----->LSB
  byte g7 = !digitalRead(pin7); //MSB
  byte g6 = !digitalRead(pin6);           // ! boolean negation
  byte g5 = !digitalRead(pin5);           // Tranzisztorok 1 értéke a logikai 0.
  byte g4 = !digitalRead(pin4);           //
  byte g3 = !digitalRead(pin3);
  byte g2 = !digitalRead(pin2);
  byte g1 = !digitalRead(pin1);
  byte g0 = !digitalRead(pin0); //LSB

  byte bin[8] = { 0 };  //bin[7] MSB------- bin[0] LSB
//---------------------------
  bin[7] = g7;
  bin[6] = bin[7] ^ g6;
  bin[5] = bin[6] ^ g5;
  bin[4] = bin[5] ^ g4;
  bin[3] = bin[4] ^ g3;
  bin[2] = bin[3] ^ g2;
  bin[1] = bin[2] ^ g1;
  bin[0] = bin[1] ^ g0;
  //---------------------------
  return ((((((bin[7] * 2 + bin[6]) * 2 + bin[5]) * 2 + bin[4]) * 2 + bin[3]) * 2 + bin[2]) * 2 + bin[1]) * 2 + bin[0];
```

**5. ábra: Programrészlet a dekódoláshoz**

## 4. Hol tartunk?

A pillanatnyi helyzet szerint kész van a program, amit lehetett, azt leteszteltünk (pl. külön az encoder beforgatását), azonban a részek összeillesztése egésszé még hátravan. Annyi tapasztalattal már rendelkezünk, hogy az eddigi tesztek eredményei csak kisebb optimizmusra adjanak alapot, még meg kell küzdenünk azért, hogy a teljes rendszer működjön.

Azonban ha összeáll minden, akkor precíz adatokkal rendelkezünk majd a napállandóról, amelynek értéke jelentősen befolyásolja mindennapjainkat és jövőnket is.