8. Távolságmérés ultrahangos HC-SR04-es szenzor segítségével

Az előző projekthez hasonlóan most is távolságot fogunk mérni és szintént centiméterekben szeretnénk azt megjeleníteni. Ebben a projektben a HC-SR04-es szenzort fogjuk használni mely ultrahang segítségével határozza meg a kívánt távolságot. Viszont most a távolság vizualizációját nem csupán a kijelzőre való kiíratás formájában fogjuk megvalósítani, hanem egy RGB led dióda kapcsolásával is. Feltétel vizsgálatokhoz kötjük, hogy az RGB diódánk mikor melyik színben világítson.

Szükséges eszközök: Számítógép, Arduino IDE, Arduino Uno, egy db 2x16 karakteres LCD kijelző I2C adapterrel, egy db HC-SR04 ultrahangos szenzor, egy db RGB led dióda, USB kábel és vezetékek.

Ennél a szenzornál nincs szükségünk könyvtár implementálására. Viszont a további header fájlokat mindenképp be kell illesztenünk, mint tudjuk ezek a *Wire.h* és a *LiquidCrystal_I2C.h*.

```
#include <Wire.h> //Wire könyvtár beillesztése az I2C busz használatához
#include <LiquidCrystal_I2C.h> //Az I2C Folyékony kristályos LCD kijelző kezelő könyvtára
LiquidCrystal_I2C lcd(0x27, 16, 2); //Az általunk használt kijelző karakterkészlete 16 karakter és 2 sor
const int trigPin = 9; //konstans globális integer típusú változó mely az Ultrahangos szenzor trigger pin-jét tárolja
const int echoPin = 10; //konstans globális integer típusú változó mely az Ultrahangos szenzor echo pin-jét tárolja
long idotartam; //long típusú változó mely az időtartam értékét tárolja
int tavolsag; //integer típusú változó mely a távolság értékét tárolja
int blue = 11; //A kék led pinjét tartalmazó globális integer változó
int green = 12; //A zöld led pinjét tartalmazó globális integer változó
int red = 13; //A piros led pinjét tartalmazó globális integer változó
int veszely; //globális integer típusú változó
void setup()
{
  lcd.init(); //Az LCD kijelző inicializálása
  lcd.backlight(); //Az LCD kijelző háttérvilágításának bekapcsolása
  pinMode(trigPin, OUTPUT); //A szenzor trigger pinje, mint digitális kimenet
  pinMode(echoPin, INPUT); //A szenzor echo pinje, mint digitális bemenet
  pinMode(blue, OUTPUT); //A kék ledet tartalmazó pin kimenetté alakítása
  pinMode(green, OUTPUT); //A zöld ledet tartalmazó pin kimenetté alakítása
  pinMode(red, OUTPUT); //A piros ledet tartalmazó pin kimenetté alakítása
  Serial.begin(9600); //A soros porton történő kommunikáció bitrátája
}
void loop() //ciklus
{
  digitalWrite(trigPin, LOW); //A trigger pin lenullázása
  delayMicroseconds(2); //Várakozás 2 mikroszekundum ideig
  digitalWrite(trigPin, HIGH); //A trigger pin aktiválása 1-es vagy HIGH értékkel
  delayMicroseconds(10); //A trigger pin aktiválása 10 mikroszekundum ideig
  digitalWrite(trigPin, LOW); //A trigger pin deaktiválása
  idotartam = pulseIn(echoPin, HIGH); //Az echo pinre érkező jel olvasása és tárolása az időtartam változóban
  tavolsag = idotartam * 0.034 / 2; //A távolság kiszámítása, valamint értékének a tárolása a távolság változóban
  Serial.println("A távolság: "); //Megadott karakterlánc soros porton való megjelenítése
  Serial.println(tavolsag); //A változó értékenek a soros porton való megjelenítése
  lcd.setCursor(3, 0); //Kurzor pozicionálás ez esetben 3. karakter a 0. sorban
  lcd.print("A TAVOLSAG:"); //Megadott karakterlánc kiíratása
  lcd.setCursor(6, 1); //Kurzor pozicionálás ez esetben 6. karakter a 1. sorban
  lcd.print(tavolsag); //A távolság változó értékének a kiíratása
  lcd.setCursor(9, 1); //Kurzor pozicionálás ez esetben 9. karakter a 1. sorban
```

1. ábra ultrahangos távolságmérés programkód
(forrás: saját szerkesztés)

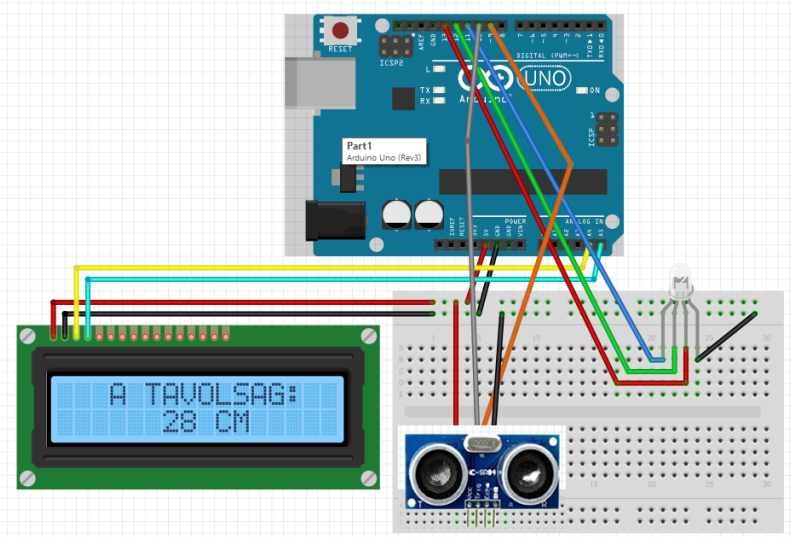
Ezen műveletek elvégzése után deklaráljunk kettő darab globális konstans integer típusú változót melyek a szenzorunk két vezérlésre szolgáló pinjéhez csatlakoznak. Az egyik a trigger a másik pedig az echo címkével van ellátva. Az Arduino Uno 9-es digitális ki- és bemenetére csatlakoztassuk a triggert a 10-es pin re pedig az echot. Továbbá hozzunk létre néhány változót, először is egy long típusút mely az időtartam értékét tárolja, egy integer típusút mely a távolságot tárolja és három darab szintén integer típust melyek az RGB led dióda kék, zöld és piros csatlakozójának megfelelő pint tartalmazzák. Végül hozzunk létre még egy szintén egész szám tárolására alkalmas változót a mi esetünkben ez a változó a *veszely* nevet kapta. Következik a programunk void setup() szekciója, ahol az lcd kijelző inicializálásával, szükség szerint tartalmának törlésével, valamint a háttérvilágítás aktiválásával kell kezdenünk. Ha ezt megtettük akkor folytatjuk az Arduino ki- illetve bemeneteink a meghatározásával. A trigger pint állítsuk be, mint digitális kimenetet az echo pint pedig, mint digitális bemenetet, mivel a trigger pin aktiválásával küldjük ki az ultrahang jelet az echo pinre pedig fogadjuk annak visszatérését. Az RGB led dióda esetében használt három változónk természetes kimenetként kell, hogy működjenek. Végül engedélyezzük a soros port kommunikációt a szokásos Serial.begin() funkcióval, a baud érték nem változik marad 9600. Tovább haladva a ciklusban találjuk magunkat itt az ultrahang kibocsátásával kezdjük. Mégpedig elsőként a trigger pint kell lenulláznunk ezt egyszerűen egy LOW értékadással tudjuk megtenni. A lenullázást követően mindössze két mikroszekundum idő után a trigger pin HIGH értékadásával kibocsáthatjuk az ultrahang jelet. Ennek az időtartama tíz mikroszekundum, ez után újból lenullázzuk ezt a kimenetet. Az echo pinre visszakapott impulzus szerű jelet az időtartamot tároló változóba helyezzük. Az impulzus megjelenésének az észlelésére a pulseIn(echoPin, HIGH) funkciót használjuk. Majd ebből eredően egy művelet segítségével kiszámolhatjuk a távolságot és a távolság értékét hordozó változónkban tárolhatjuk is azt. Ezt követi a távolság kiíratása először a soros portra majd az lcd kijelzőre is egyaránt. A kijelző mutatóját a 0-ik sor 3-ik karakterére állítjuk és a következő karakterláncot jelenítjük meg ”A TÁVOLSÁG:” majd a kurzort az 1-es sor 6-ik karakterére állítjuk és a megfelelő változónk értékét is kiíratjuk az lcd.print(tavolsag) funkció segítségével.

```
  lcd.setCursor(9, 1); //Kurzor pozicionálás ez esetben 9. karakter a 1. sorban
  lcd.print("CM"); //Megadott karakterlánc kiíratása
  if (tavolsag > 150) //Szelekció ha a mért távolság nagyobb, mint 150 cm akkor:
  {
    veszely = 3; //A változó értéke = 3
  }
  if (tavolsag < 100 & tavolsag != 50) //Szelekció ha a mért távolság kisebb, mint 100 cm és nem egyenlő 50 cm akkor:
  {
    veszely = 0; //A változó értéke = 0
  }
  if (tavolsag < 45 & tavolsag != 25) //Szelekció ha a mért távolság kisebb, mint 45 cm és nem egyenlő 25 cm akkor:
  {
    veszely = 1; //A változó értéke = 1
  }
  if (tavolsag < 20) //Szelekció ha a mért távolság kisebb, mint 20 cm akkor:
  {
    veszely = 2; //A változó értéke = 2
  }
  switch(veszely) //Switch case többirányú elágazás:
  {
    case 0: //Ha a veszély változó értéke = 0 akkor:
      digitalWrite(blue, HIGH); //Kék led világít
      break; //Kilépés az elágazásból
    case 1: //Ha a veszély változó értéke = 1 akkor:
      digitalWrite(green, HIGH); //Zöld led világít
      digitalWrite(blue, LOW); //Kék led nem világít
      break; //Kilépés az elágazásból
    case 2: //Ha a veszély változó értéke = 2 akkor:
      digitalWrite(red, HIGH); //Piros led világít
      digitalWrite(blue, LOW); //Kék led nem világít
      digitalWrite(green, LOW); //Zöld led nem világít
      break; //Kilépés az elágazásból
    default: //Egyébként
      digitalWrite(blue, LOW); //Kék led nem világít
      digitalWrite(green, LOW); //Zöld led nem világít
      digitalWrite(red, LOW); //Piros led nem világít
      break; //Kilépés az elágazásból
  }
  delay(500); //Várakozás 0.5 másodpercig
  lcd.clear(); //Az LCD kijelző tartalmának a törlése
}
```

2. ábra ultrahangos távolságmérés programkód
(forrás: saját szerkesztés)

Tovább haladva a megjelenített távolság érték mellé írassuk ki a „CM” karakterláncot is, persze előtte a kurzort a megfelelő pozícióba kell állítanunk, ha szeretnénk tudni, hogy körülbelül hány digitet nagyságú a mért távolság értéke akkor azt leellenőrizhetjük a soros portunkon, mivel előzőleg ezt az értéket már megjelenítettük ott is. Következhet az RGB led dióda feltétel vizsgálathoz kötött kivilágítása. Elsőként a kék színnel kezdjük a mi esetünkben ezzel a színnel fogjuk jelölni a legnagyobb távolságot, ami a jelen esetben nagyobb, mint 150 centiméter. Ezért a feltételt a következőképpen írjuk meg *if(tavolsag > 150)*. Viszont a megszokottakkal szemben most egy kicsit más módon fogunk eljárni. A led dióda világításának aktiválását nem a feltétel vizsgálat igaz ágában fogjuk elvégezni, hanem egy un. *Switch* több irányi elágazást fogunk használni. Ennek a lényege, hogy van egy változónk jelen esetben ez a változó a *veszely* nevezetű integer típusú és globális változónk. A *Switch*-ről azt kell tudni, hogy több ággal rendelkezik, ezeket a *Case*-parancsal tudjuk létrehozni és a *Break* parancsal tudunk belőlük kilépni. Minden *Case* rendelkezik egy sorszámmal és ez a sorszám megfelelhet a vizsgált változó értékének is. Tehát ha a *veszely* változó értéke nulla akkor a *Switch* a nulla sorszámú *Case*-ben található

műveleteket fogja lefuttatni. A mi helyzetünkben a *Switch* három darab *Case*-szel rendelkezik, tehát ha a változónk 0, 1 vagy 2 es értékkel bír akkor az annak megfelelő *Case* kerül lefuttatásra, viszont, ha a változónk egyéb értékkel bír akkor a *Switch* ún. *Default* ága fut le, akárcsak az *Else* egy általános feltétel vizsgálatnál. Így tehát ha a változónk értéke nulla akkor az RGB led dióda kék színben világít, 1-es érték esetén a szín zöldre vált és végül 2-es értéknél pedig vörös színben világít. A *Switch Default* ágában pedig a led dióda kikapcsolt állapotban marad. Végül elhelyezünk egy késleltetés funkciót melynek értékét 500 milliszekundumra állítjuk, majd töröljük az lcd kijelzőnk tartalmát, így a ciklus minden egyes ujraindulásánál a kijelzőnk tartalma a szenzortól kapott aktuális értéket fogja mutatni.



3. ábra ultrahangos távolságmérés bekötési rajz
(forrás: saját szerkesztés)

Az áramkör megépítéséhez vegyük szemügyre a fenti 3.ábrát itt láthatjuk, hogy pontosan mit és hogyan kell csatlakoztatnunk. Láthatjuk, hogy a kijelzőnk adat és tápellátása nem változott az előző projektekhez képest. A HC-SR04-es szenzorunk Vcc pinjére csatlakoztassuk az Arduino Uno 5 voltot biztosító pinjét, valamint a GND pint pedig csatlakoztassuk a szenzor Gnd-vel jelzett pinjéhez. Ami a trigger és echo pineket illeti a programkódból tudhatjuk, hogy ezek a 9-es és 10-es pineken foglalnak helyet. Végül az RGB led diódát is kössük be a következő módon, a B-vel jelzett lábat az Arduino 11-es pinjére a G-vel jelzettet a 12-es pinre és az R-rel jelzettet pedig a 13-as pinre.

Majd csatlakoztassuk az USB kábelt az Arduino csatlakozójába és töltsük fel a programot. Amennyiben a *Done uploading* üzenetet kapjuk a feltöltés sikerrel járt és az áramkör a következőképpen működik.