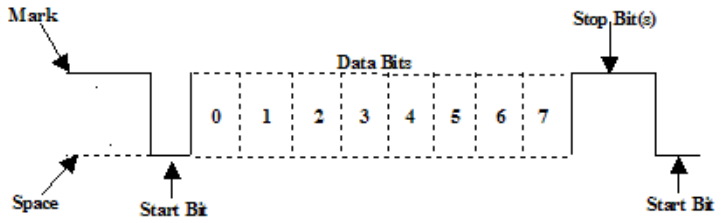
Arduino Uygulama Çalışması-5

Serial Monitör ve LDR ile Ortam Işık Kontrolü

Seri İletişim : Seri haberleşme iletilecek sinyallerin tek hat üzerinden ard arda(bit-bit) gönderilmesi demektir. Seri haberleşmeye örnek olarak hemen hemen tüm bilgisayarlarda bulunan USB, RS232 ve Ethernet portunu gösterebiliriz.



RS232 portu ile seri haberleşmede bir veri gönderme ve bir de veri alma hattı olarak iki hat kullanılır. Arduino UNO R3 üzerinde bu pinler 0 numaralı dijital pininin özelliği olan RX veri alma ve 1 numaralı dijital pinin özelliği olan TX veri gönderme pini üzerinden sağlanır.

Kartımız üzerinde USB seri port dönüştürücü chip bulunduğu için direkt olarak USB kablosu üzerinden bilgisayar ile seri haberleşme sağlayabiliriz.

0 ve 1 nolu dijital pinler aynı anda hem dijital pin hem de seri iletişim pini olarak kullanılamazlar. Aynı pinler programlama için de kullanıldığından bu pinlere bağlı seri iletişim cihazları var ise bu cihazı arduino cihazımızdan ayırmadan program yüklemeye çalışırsak hata mesajı verecektir.

Seri iletişimde birtakım kurallar vardır. Bu kurallardan bir tanesi BaudRate diye bilinen seri haberleşme hızıdır. Bilgiler ard arda gönderildiği için, arduino ile seri iletişim yaptığı cihazın 1 saniyedeki veri gönderme-alma hızı aynı olmak zorundadır. Bu işlemi programımızın setup() yapısı içerisinde ***Serial.begin()*** fonksiyonu ile tanımlarız ve parantezler içerisine haberleşme hızını integer cinsinden değerini yazarız. Böylece seri iletişim pinlerini başlatmış oluruz.

```
void setup() {
     Serial.begin(9600); // saniyede gönderilecek bit sayısı 9600 bit
                         //olarak ayarlandı
}
```

Serial.begin();

Bu fonksiyonumuz yeni bir seri haberleşme tanımlamak için kullanılır. Serial.begin() fonksiyonu Arduino UNO için 0 ve 1 numaralı pinlerinde bulunan fiziksel seri haberleşme özelliğini aktif eder.

Şimdi Serial.begin() fonksiyonun alabileceği parametrelere bakalım;

Sözdizimi: **Serial.begin**(**baudRate**);

baudRate : Bu özellik olmazsa olmaz olan seri portun saniyede göndereceği ve alacağı bit sayısını belirler. Seri portun haberleşeceği cihaz ile aynı olmak zorundadır. Eğer aynı olmaz ise göndereceğiniz ya da alacağınız verilerin haberleşme yaptığınız cihaz ile çok farklı olduğunu görürsünüz. Çoğu cihazın

seri haberleşme hızı 9600 Baud Rate'dir. Bu değer cihaz saniyede 9600 bit gönderip alabilir demektir. 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 değerleri alabilir.

Serial.print() Fonksiyonu:

Bu fonksiyon gönderilecek olan verinin ASCII standartlarında gönderilmesini sağlar. Yani veriyi karşıya direkt karakter olarak gönderir. Bu fonksiyon ile gönderilen veriler makina tarafından ekrana basılabilir ve insan tarafından okunabilir.

Sözdizimi; **Serial.print**(**gonderilecekVeri**);

```
Serial.print(78); // "78" gönderir

Serial.print(1.23456); // "1.23" gönderir

Serial.print('N'); // "N" gönderir

Serial.print("Hello world."); // "Hello world." gönderir
```

Direkt olarak girilen karakterleri seri porta gönderse de ikinci bir parametre olarak gönderilecek verinin formatı belirlenebilir;

Serial.print(**veri**, **format**);

İzin verilen format değerleri; BIN (Binary yani 2 tabanlı veri), OCT (Octal yani 8 tabanlı veri), DEC (Decimal yani 10 tabanlı veri) veya HEX (Hexadecimal yani 16 tabanlı veri) olarak gönderilebilir.

```
Serial.print(78, BIN); // "1001110" gönderir

Serial.print(78, OCT); // "116" gönderir

Serial.print(78, DEC); // "78" gönderir

Serial.print(78, HEX); // "4E" gönderir
```

Ondalıklı sayılar için ise ikinci parametre noktadan sonra gösterilecek basamak sayısını belirleyebilir;

```
Serial.print(1.23456, 0); // "1" gönderir

Serial.print(1.23456, 2); // "1.23" gönderir

Serial.print(1.23456, 4); // "1.2346" gönderir
```

Serial.println() Fonksiyonu:

Serial.println() fonksiyonu Serial.print() fonksiyonu ile tamamen aynı işlemi yapar, Ancak bu fonksiyon gönderdiği her verinin sonuna Satır Başı (ASCII 13 veya '\r') karakteri ve ardından Yeni Satır(ASCII 10 veya "\n") karakterini ekler. Böylelikle gönderilen her veride alıcı cihaz alt satırdan devam eder.

Sözdizimi; **Serial.println**(**gonderilecekVeri**);

```
Serial.println("Merhaba"); // "Merhaba" yaz ve alt satıra geç
```

Serial.available() Fonksiyonu:

Serial.available() fonksiyonu seri port arabelleğindeki (buffer) alınan byte sayısını verir. Biz seri porta gelen veriyi almadan önce bu veriler seri port arabelleğine kayıt edilir. Seri port arabelleği 64 byte değerindeki veriyi saklayabilecek kapasitededir. Biz Serial.available() fonksiyonu ile gelen verinin kaç byte'dan oluştuğunu öğrenebiliriz. Fonksiyon, herhangi bir parametre almaz ve gelen verinin byte sayısını integer cinsinden değer olarak bize verir.

Sözdizimi: Serial.available();

Bu fonksiyonu seri porta veri gelip gelmediğini anlamak için de kullanabiliriz.

Örneğin seri porta veri geldiğinde, gelen verileri alan program yazalım;

```
void loop() {
  if(Serial.available()>0){ // Eğer seri porta veri gelmişse
    int gelen = Serial.read(); // gelen veriyi "gelen" değişkenine kaydet
  }
}
```

Serial.read() Fonksiyonu:

Seri Porttan gelen veriyi ilk byte ile başlayarak sırayla okuma işlemini gerçekleştirir ve okuduğu her değerin integer karşılığını döndürür. Bu integer değer tip dönüşümü yapılarak istenilen formda kullanılabilir. Eğer okunacak veri yoksa -1 değerini döndürür.

```
int alinan = 0; // alınacak degeri saklamak için kullanılacak değişken

void setup() {
  Serial.begin(9600); // Seri Portumuzu 9600 baud olarak ayarlıyoruz.
}

void loop() {
  if (Serial.available()) { // Serial Porta kontrol et
                            // eğer girdi varsa oku ve yazdır.

    alinan = Serial.read(); // Serial Porttan değer okuma
    Serial.print("Alinan Deger: ");
    Serial.print(alinan); // integer olarak alınan değeri yazdırma
    Serial.print(" - Char Olarak Alinan Deger: ");
    Serial.println((char)alinan); // char'a dönüştürerek alınan değeri yazdırma.
  }
}
```

Serial.readBytes() Fonksiyonu :

readBytes fonksiyonu Serial Porttan gelen veriyi istenilen byte adedinde okuyarak char veya byte tipinde bir diziye yazma `işlemini` gerçekleştirir. Bu işlemi gerçekleştirirken geriye kaç byte lık işlem yaptığını integer tipinde döndürür.

Serial.readBytes(buffer, length)

buffer : char veya byte tipinde bir dizi
length : integer tipinde kaç byte lık okuma yapılacağı

```
int byteSayisi = 0;
char dizi[10]; // 10 elemanlı bir dizi oluştur.
void setup() {
  Serial.begin(9600); // Seri Portu başlat.
}

void loop() {

  if (Serial.available() > 0) { // Serial kontrol et
  byteSayisi = Serial.readBytes(dizi,4); // 4 byte okuma yap ve diziye yaz.
                                         //Ayrıca gelen verinin kaç byte inin işlendiğini
                                         //sakla.

    Serial.print("Dizi : ");
    Serial.println(dizi); // üzerine yazılmış olan diziyi ekrana yaz.
    Serial.println(byteSayisi); // kaç byte lık işlem yapıldığını yaz.
  }
}
```

Serial.readBytesUntil () Fonksiyonu :

Bu fonksiyon **readBytes** fonksiyonundan farklı olarak bir char parametre daha alır ve bu aldığı char parametre gelen veri dizisi içerisinde varsa okuma işlemini sonlandırır.

Serial.readBytesUntil(character,buffer, length)

character : aramak için char tipinde karakter
buffer : char veya byte tipinde bir dizi
length : integer tipinde kaç byte lık okuma yapılacağı

```
int byteSayisi = 0;
char dizi[15]; // 15 boyutlu bir dizi oluştur.
void setup() {
  Serial.begin(9600); // Seri Portu başlat.
}

void loop() {

  if (Serial.available() > 0) { // Serial Port kontrol et
    byteSayisi = Serial.readBytesUntil('.', dizi, 15);
    // En fazla 15 bytelık okuma, yada '.' gelince okumayı sonlandır ve okunan byte
    // sayısını değişkene ata.

    Serial.print("Dizi : ");
    Serial.println(dizi); // üzerine yazılmış olan diziyi ekrana yaz.
    Serial.println(byteSayisi); // kaç byte lık işlem yapıldığını yaz.

    for (int i = 0; i < 15; i++) { // diziyi sıfırla
      dizi[i] = NULL;
    }
  }
}
```

Serial.readString () Fonksiyonu :

Bu fonksiyon Serial Porta gelen girdi dizisini satır sonuna kadar okuyarak okunan değeri **String** formunda geri döndürür.

```
String okunan; // girilecek parametreyi saklamak için String değişken
void setup() {
  Serial.begin(9600); // Seri Portu başlat.
}
```

```
void loop() {

  if (Serial.available() > 0) { // Serial kontrol et
    okunan = Serial.readString(); // Serial Porttaki girdi değerini string e ata.
    Serial.println(okunan); // okunan değeri ekrana yazdır.
  }
}
```

Serial.readStringUntil ()

Bu fonksiyon **readString** fonksiyonundan farklı olarak bir char parametre daha alır ve bu aldığı char parametre gelen veri dizisi içerisinde varsa okuma işlemini sonlandırır.

```
String okunan; // girilecek parametreyi saklamak için String değişken
void setup() {
  Serial.begin(9600); // Seri Portu başlat.
}

void loop() {

  if (Serial.available() > 0) { // Serial Port kontrol et
    okunan = Serial.readStringUntil('.');
    // Serial Porttaki girdi değeri okunur ardından string e atanır
    // ve eğer . (nokta) karakterine rastlanırsa okuma sonlandır.
    Serial.println(okunan); // okunan değeri ekrana yazdır.
  }
}
```

Serial.setTimeout () Fonksiyonu :

Bu fonksiyon ile Seri Porttan gelecek olan verinin maksimum ne kadar bekleneceği mili saniye cinsinden ayarlanır. readBytes**,** readBytesUntil**,** parseFloat **veya** parseInt **gibi fonksiyonların varsayılan bekleme süreleri 1000 ms'dir.**

Serial.write () Fonksiyonu :

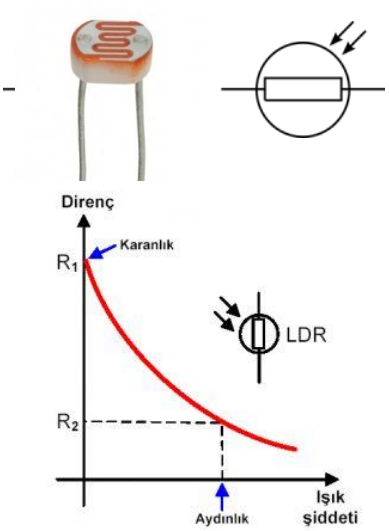
Bu fonksiyon **print** fonksiyonuna benzer bir işlev görür. Tek byte büyüklüğündeki değerleri yazdırmakta, String değerler ve char veya byte tipinde dizileri yazdırmakta kullanılır. Alacağı ikinci bir parametre ile de yazdırılan String veya dizinin kaç elemanının yazdırılacağı ayarlanabilir. Ayrıca **write** fonksiyonu geri dönüş değeri olarak üzerinde işlem yapılan byte sayısı döndürür.

```
char dizi[] = {'.','c','o','m'}; // char tipindeki dizi
void setup() {
  Serial.begin(9600); // Seri Portu başlat.
}

void loop() {

  Serial.write(77); // 77 = M yazdır.

  Serial.write("2019-20 - Arduino Dersleri",9);
  // String'in ilk dokuz karakterini yazdır.

  int byteSayisi = Serial.write(dizi); // kaç byte lık işlem yapıldığını sakla
  Serial.print("\ndizi'den okunan byte sayisi: ");
  Serial.println(byteSayisi); // kaç byte lık işlem yapıldığını yazdır.

  delay(10000);
}
```

UYGULAMA :

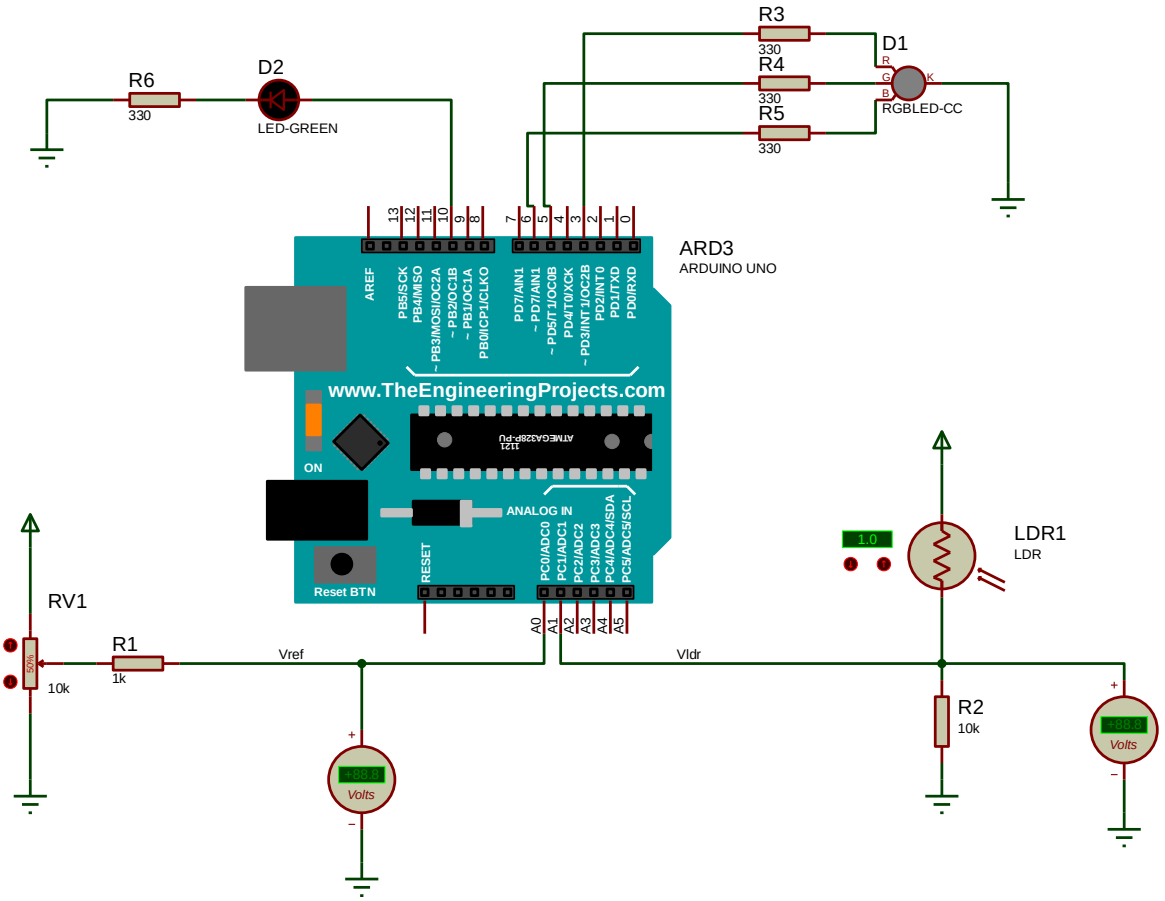
Bu çalışmada istenilen ayar değerine göre ortam ışığını otomatik olarak ayarlayabilen bir proje yapılacaktır. Ayar değeri bir potansiyometre yardımıyla belirlerken, ortam ışığının miktarını LDR denilen ışığa duyarlı dirençten oluşan bir algılama devresi, ile algılamaktayız. Ayar değeri dış ortamın değeri olarak belirlenirse yapılacak olan bu çalışma dış ortam ışık miktarına göre iç ortamın ışık miktarını otomatik olarak ayarlama işlemi yapar.

LDR(Foto direnç) Nedir?



LDR (Light Dependent Resistor), Türkçede "Işığa Bağımlı Direnç" anlamına gelmektedir. Bir diğer adı da foto dirençtir. LDR her ne kadar bir direnç çeşidi olsa da aynı zamanda pasif bir sensördür. LDR'ler bulundukları devrelerde değişen direnç değerleri ile bir çıkış sağlarlar fakat bu işlemi dış ortamdan aldıkları fiziksel bir değişim ile gerçekleştirdiklerinden dolayı bir sensör görevi görmüş olurlar.

Üzerine düşen ışık şiddeti ile ters orantılı bir çalışma prensibine sahiptir. Yani üzerine düşen ışık şiddeti arttıkça sahip olduğu direnç değeri azalır, ışık şiddeti azaldıkça sahip olduğu direnç değeri artar. LDR'ler sahip oldukları direnç değerlerinin değişmesi ile bir anahtarlama görevi görürler. Başka bir açıdan bakacak olursak bir optik sensör görevi de görmüş olurlar. Aşırı ısı altında bozulmaktadırlar (Maksimum: 60 °C).

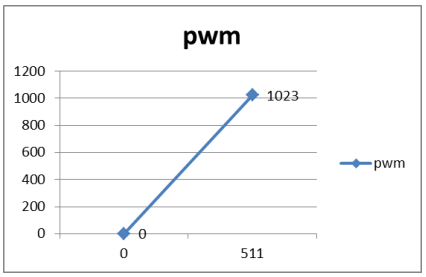


LDR ye Bağlı Olarak LED Işığının Ayar Değere Göre Kontrolü İşlem Analizi :

Ayar Deger= **512**

Ayar Deger		LDR Değer		Fark= ldr-ayar	
min	max	min(Aydınlık)	max (Karanlık)	min	max
0	1023	512	1023	0	511

fark		AN Çıkış(PWM)	
min	max	min	max
0	**511**	0	**1023**



Oransal Katsayı :

Kp **2,00196**

Kp= Max_pwm/(fark_max - fark_min)

fark= ldr-ayar

PWM çıkış = Kp*fark

```
//Seri monitör ve LDR ile ışık kontrolü kodları
#define AN_AYAR_PIN A0
#define AN_LDR_PIN  A1

#define LED_PIN   10
#define R_LED_PIN 3
#define G_LED_PIN 5
#define B_LED_PIN 6

#define ORANSAL_CARPAN  2.00    //kp

int ayar_deger = 0;
int an_ldr = 0;
int fark = 0, pwm_cikis=0;

void setup() {
  Serial.begin(9600);

  pinMode(LED_PIN, OUTPUT);
  pinMode(R_LED_PIN, OUTPUT);
  pinMode(G_LED_PIN, OUTPUT);
  pinMode(G_LED_PIN, OUTPUT);

  Serial.println("basla...");

}

void loop() {

  ayar_deger = analogRead(AN_AYAR_PIN);
  ayar_deger=map(ayar_deger,0,1023,0,255);  // 0-255 değer aralığına göre
                                            //düzenle.
  analogWrite(LED_PIN,ayar_deger);        //ayar değer parlaklığını led ile göster.
```

```
  an_ldr = analogRead(AN_LDR_PIN);  // Ortam ışık değerini oku.
  fark = an_ldr - ayar_deger;     // olması gereken değer ile arasındaki farkı bul.
  pwm_cikis=(int)(ORANSAL_CARPAN*fark);   // Çıkış ledlerine uyuglanacak değer.
  pwm_cikis=map(pwm_cikis,0,1023,0,255);  //0-255 değer aralığına göre düzenle
  analogWrite(R_LED_PIN,pwm_cikis);       // Çıkış ledlerine değeri yaz.
  analogWrite(G_LED_PIN,pwm_cikis);
  analogWrite(B_LED_PIN,pwm_cikis);
  //Elde edilen ve hesaplanan değerleri seri monitörde göster.
  Serial.print("Ayar Deger:"); Serial.println(ayar_deger);
  Serial.print("LDR:"); Serial.println(an_ldr);
  Serial.print("Fark:"); Serial.println(fark);
  Serial.print("PWM:"); Serial.println(pwm_cikis);

  delay(500);   //işlem peryodu

}
```