

Arduino programlama



Önsöz

iki bölümden oluşan bu kitapta birinci bölüm Arduino nedir, ne değildir ve programlama için referans kısmından oluşurken ikinci kısım ise Arduino ile yapılmış projelerden oluşuyor. Kitabın amacı Arduino'ya yeni başlayanlar için basit anlaşılır bir kaynak sunmak.

Lisans

Bu belgeyi, Open Publication Licence lisansının 1.0 ya da daha sonraki sürümünün koşullarına bağlı kalarak kopyalayabilir, dağıtabilir ve/veya değiştirebilirsiniz. Bu Lisansın özgün kopyasını <http://www.opencontent.org/openpub/> adresinde bulabilirsiniz.

Not: Burdaki notlarla ilgili gördüğünüz bir sorun ve önerileriniz için nazrdogan@gmail.com mail atabilirsiniz.

İçindekiler nedir?

1. Fiziksel Programlama nedir?
2. Arduino Nedir?
3. Neden Arduino?
4. Arduino çeşitleri ve Shield'leri
5. Arduino Kurulum

Programlama

Temel

1. Arduino Programlama dilinin Kod yapısı
2. Minimum Arduino Sketch'i
3. ilk Arduino Programı
4. Referanslar

Projeler

1. proje
2. proje
3. proje
4. proje
5. proje
6. proje
7. proje
8. proje
9. proje
- 10.proje

Nazır DOĞAN

www.gereksizcoder.wordpress.com

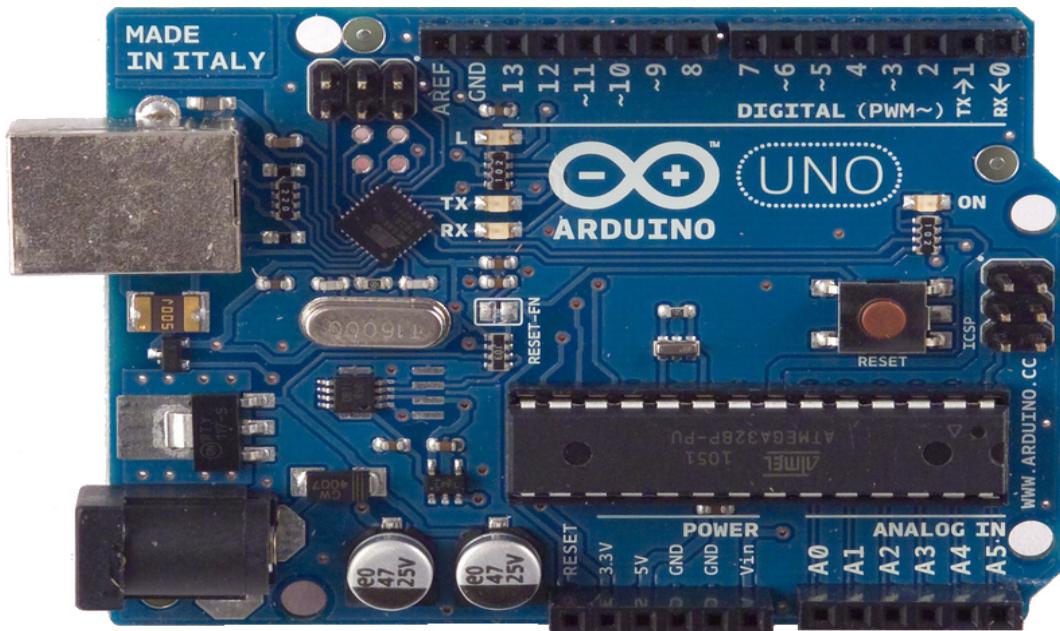
V.0.1

Fiziksel Programlama

Fiziksel programlama yazılım ve donanım kullanarak analog dış dünyayla veri alışverişi yapan fiziksel tasarlama işine verilen isimdir. Sensorler yardımıyla dış dünyayla iletişime geçilir. Analog olan veriler dijitalde aktarılır ve yazılım sayesinde ne yapılacağına karar verilir.

Arduino Nedir?

Arduino fiziksel dünyayı algılayan ve kontrol edebilmek için kullanabileciginiz basit bir bilgisayardır. Basit bir mikroişlemci ve yazılım yazmak için bir geliştirme ortamına sahip basit bir fiziksel hesaplama platformudur.



Arduino anahtarları ,sensorleri,motorları ve diger fiziksel çıkışları kontrol etmek ve etkileşimli nesneler geliştirmek için kullanılabilir. Arduino projeleri tek başına geliştirebilir yada bilgisayar üzerinde çalışan yazılımlara baglanabilir .
(Flash,Processing vb..)

Arduino açık kaynak bir geliştirme ortamına sahip olup Processing yazılımından yola çıkararak yazılmıştır.

Kendi board'unuzu yapabileceginiz gibi tamamen hazır alabilirsiniz.ve tamamen açık kaynak kodlu bir IDE'ye sahiptir

Neden Arduino ?

Fiziksel ortam için birçok mikrokontrolör ve mikrokontrolör platformu mevcuttur. Örnegin Parallax Basic Stamp,Netmedia's BX-24,Phidgets,MIT's Handyboard vs.. birçogu aynı fonksiyona sahiptir. Fakat hepsinin programlanması oldukça zordur.Arduino ise programlamayı oldukça kolay hale getirir.Öğrenciler ve amatörler için büyük avantajlar saglar.

Nedenler ise

- Ucuz olması-Arduino diger platformlarla karşılaşığında daha ucuzdur.ayrıca kendiniz yapabilirsiniz.
- Çapraz platform olması-Arduino Linux,Windows ve MacOs ta çalışabilir.Çogu mikrokontrolör sistemi Windows'la sınırlıdır.
- Basit ve Açık Programlama Ortamı-Arduino yazılımı yeni başlayanlar için oldukça kolay bir ortam sunar.
- Açık Kaynak olması-Benim düşüncem en önemli sebeptir.Gereksiz bir çok yazılıma para vermekten kurtarır ve devamlı gelişebilen bir ortam sunar.

Arduino Çeşitleri ve Shield'leri

Arduino UNO

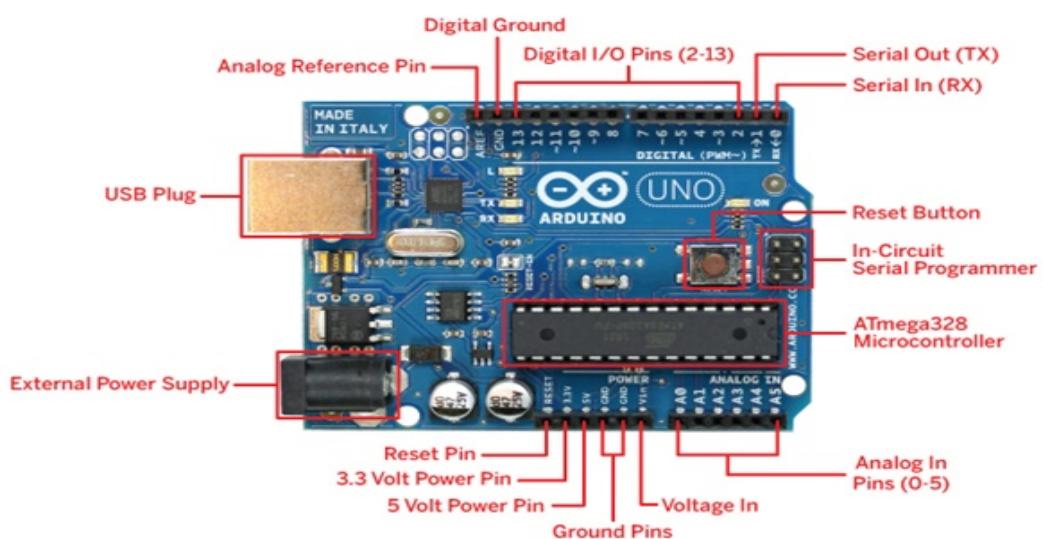
Arduino Uno ATMEGA 328 temelli bir board'dur. 14 tane dijital giriş/çıkış (G/Ç) (6 tanesi PWM için kullanılabilir) pinine sahiptir. 6 tane analog giriş ve 16 Mhz seramik resonator'e bulunmaktadır. USB bağlantısı sayesinde kolayca yazdığınız kodu yükleyip çalıştırabilirsiniz. Aynı zamanda USB bağlantısını güç içinde kullanabilirsiniz.

Ya da üzerinde bulunan Güç girişi ile bir güç kaynağına bağlayabilirsiniz. Bir adette ICSP bağlantısı bulunmaktadır.
Özet olarak

Mikrokontroller	ATMEGA 328
Çalışma Voltajı	5V
Giriş Voltajı (tavsiye edilen)	7-12V
Giriş Voltajı (limit)	6-20V
Dijital G/Ç	14 (6 tanesi PWM destekliyor)
Analog Girişler	6
DC akım her pin'de	40mA

3.3V pin için DC akım	50mA
Flash Hafıza	32KB
SRAM	1KB
EEPROM	1KB
Saat Hızı	16MHz

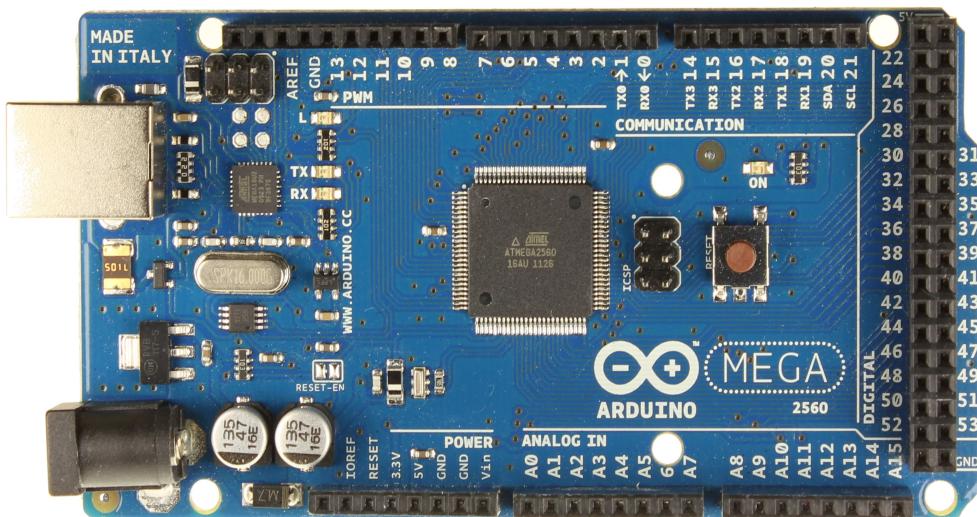
Arduino Uno'dan bahsettikten sonra Uno'nun önemine



gelelim.Uno,Arduino programlamaya yeni başlayanlar için
biçilmiş kaftandır.Hem diger board'lara nazaran ucuz oluşu
her yerde bulunabilir olması ve Arduino shield'lerin hemen
hemen hepsiyle uyumlu olması nedeniyle tercih sebebidir.

Arduino MEGA 2560

Arduino Mega 2560 ATMEGA 2560 temelli bir board'dur. 54 tane dijital G/Ç (15 tanesi PWM için kullanılabilir). 16 tane analog giriş ve 4 UART(donanım seri portu) bulunmaktadır. 16 MHz kristal osilator, USB bağlantı ve güç girişine sahiptir. ICSP bağlantısı ve reset butonu bu board'da mevcuttur.



Özet olarak

Mikrokontroller

ATMEGA 2560

Çalışma Voltajı

5V

Giriş Voltajı (tavsiye edilen)

7-12V

Giriş Voltajı (limit)

6-20V

Dijital G/Ç

54(15 tanesi PWM destekliyor)

Analog Girişler	6
DC akım her pin'de	40mA
3.3V pin için DC akım	50mA
Flash Hafıza	256KB
SRAM	8KB
EEPROM	4KB
Saat Hızı	16MHz

Arduino Mega'nın özelliklerinden bahsettiğten sonra ise Mega neden önemli bir boarddur ondan bahsedelim.

Arduino Mega hafıza ve pin sayısından dolayı daha büyük projeler için ideal bir boarddur .Uno'nun yetmediği projeler için kullanılabilir.

Mega,Uno için tasarlanmış bir çok shield ile uyumludur.

Diger Arduino Board'ları

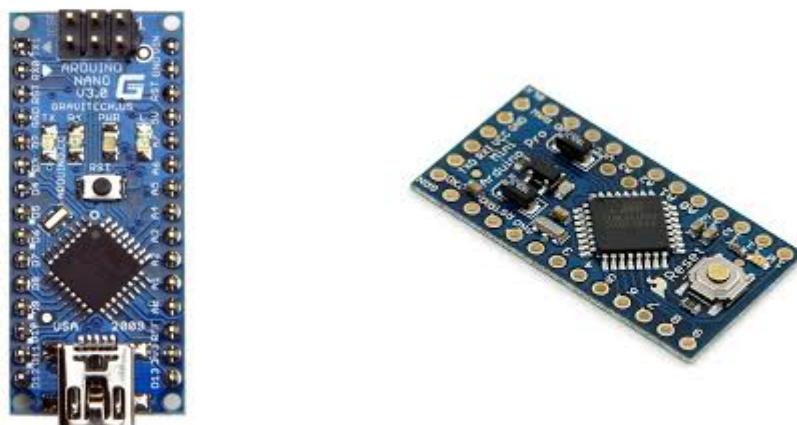
Burada bütün Arduino Boardlarından bahsetmem çokta mümkün degil Muhtemelen ben bu yazıyı yazarken yeni bir çok board tasarılmaktadır . :)

Sadece isimlerini ve birkaç özelliğini yazarak geçmek istiyorum.

İlk ve tek 32 bit Arduino DUE



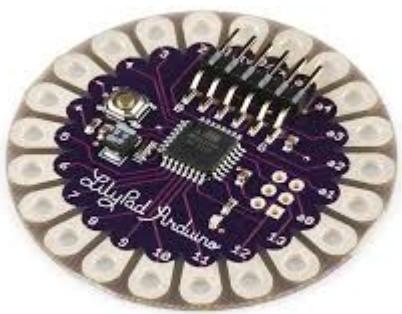
Arduino Nano ve Mini



Arduino ESPLORA ve Leonardo



Lilypad



Shield'ler

Shield'leri tanıtmadan önce shield nedir onun üzerinde biraz durmak istiyorum.Shield'ler Arduino boardumuzun üzerine kolayca takılıp yapmak istediğimiz projeyi dahada kolay yapmamızı saglayan ek board'lardır.

Arduino'nun resmi olarak birçok shield'i bulunmaktadır.ama açık kaynak olmasından dolayı geliştiriciler kendi

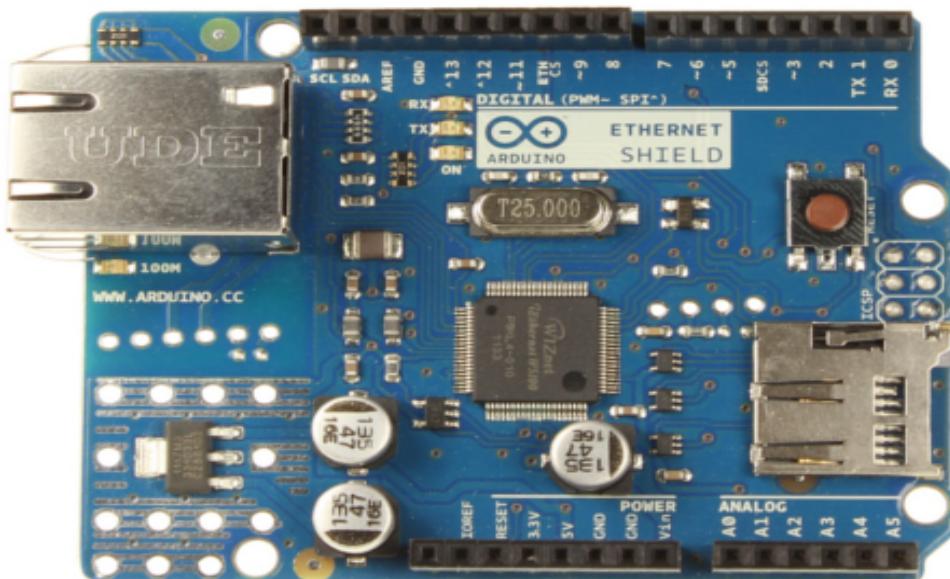
shield'lerini tasarlayıp satmakta veya kullanmaktadır.

Resmi Arduino Shield'lerinden Bazıları

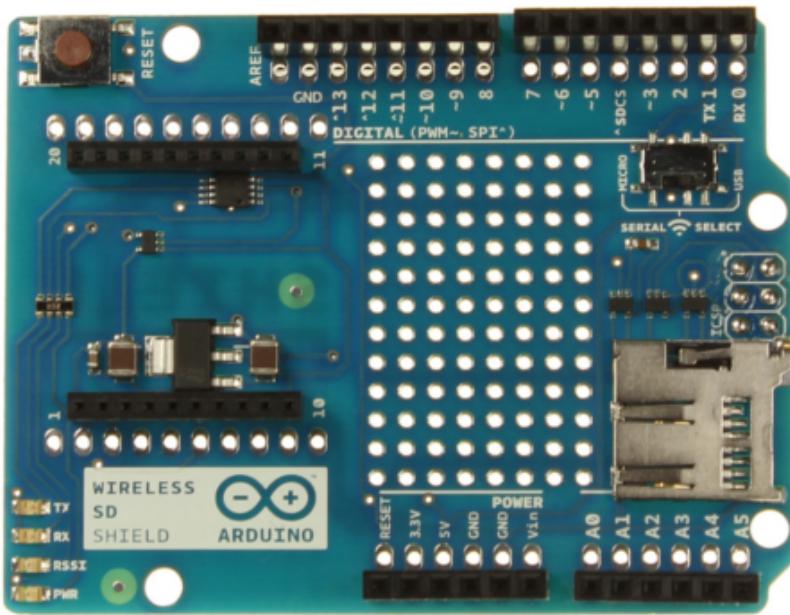
GSM shield



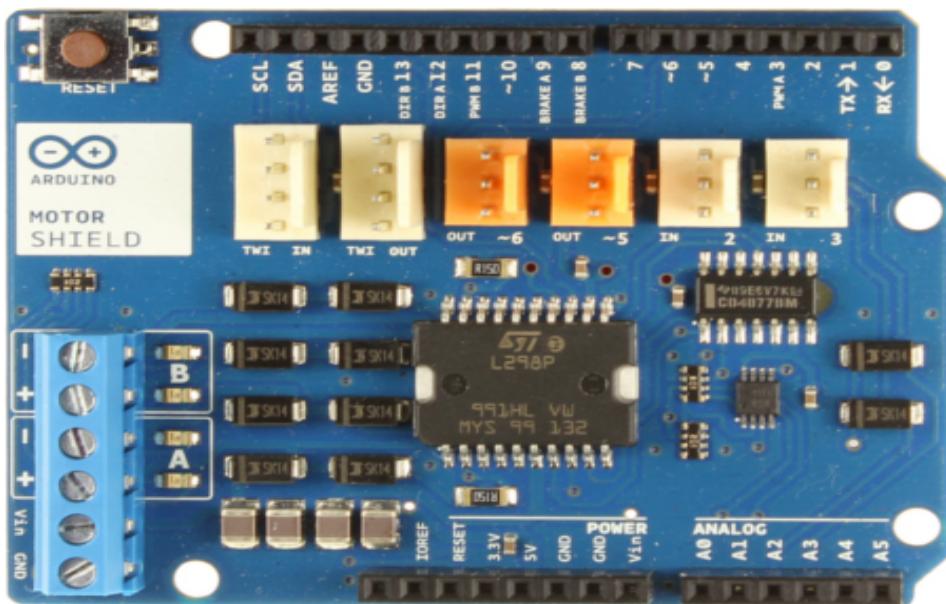
Ethernet Shield



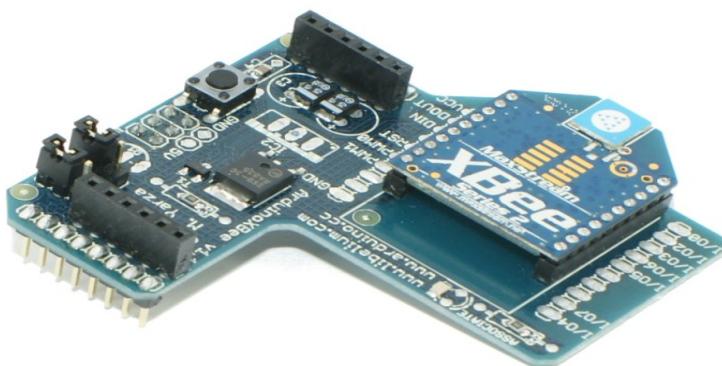
Wireless SD Shield



Motor Shield



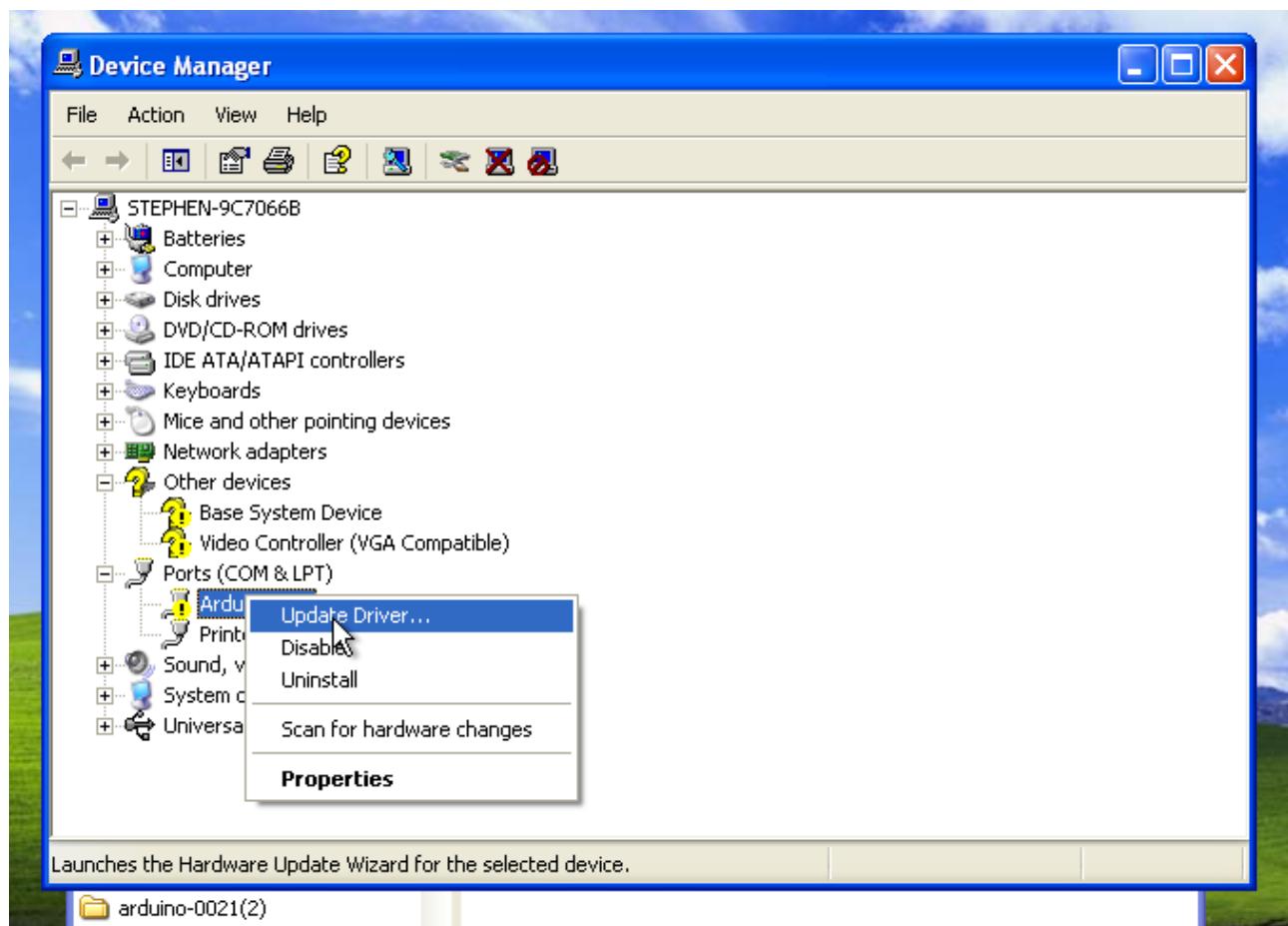
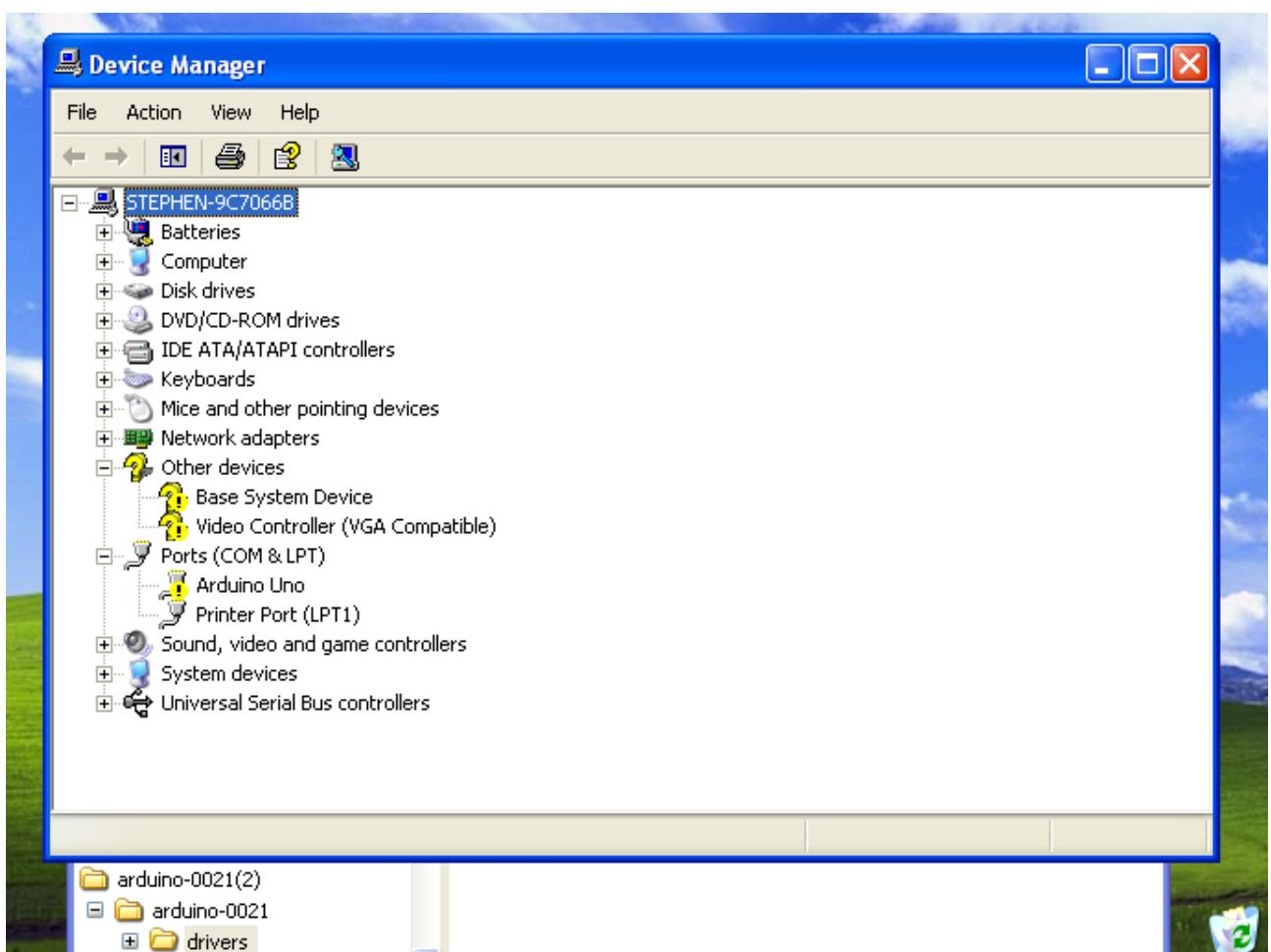
Xbee

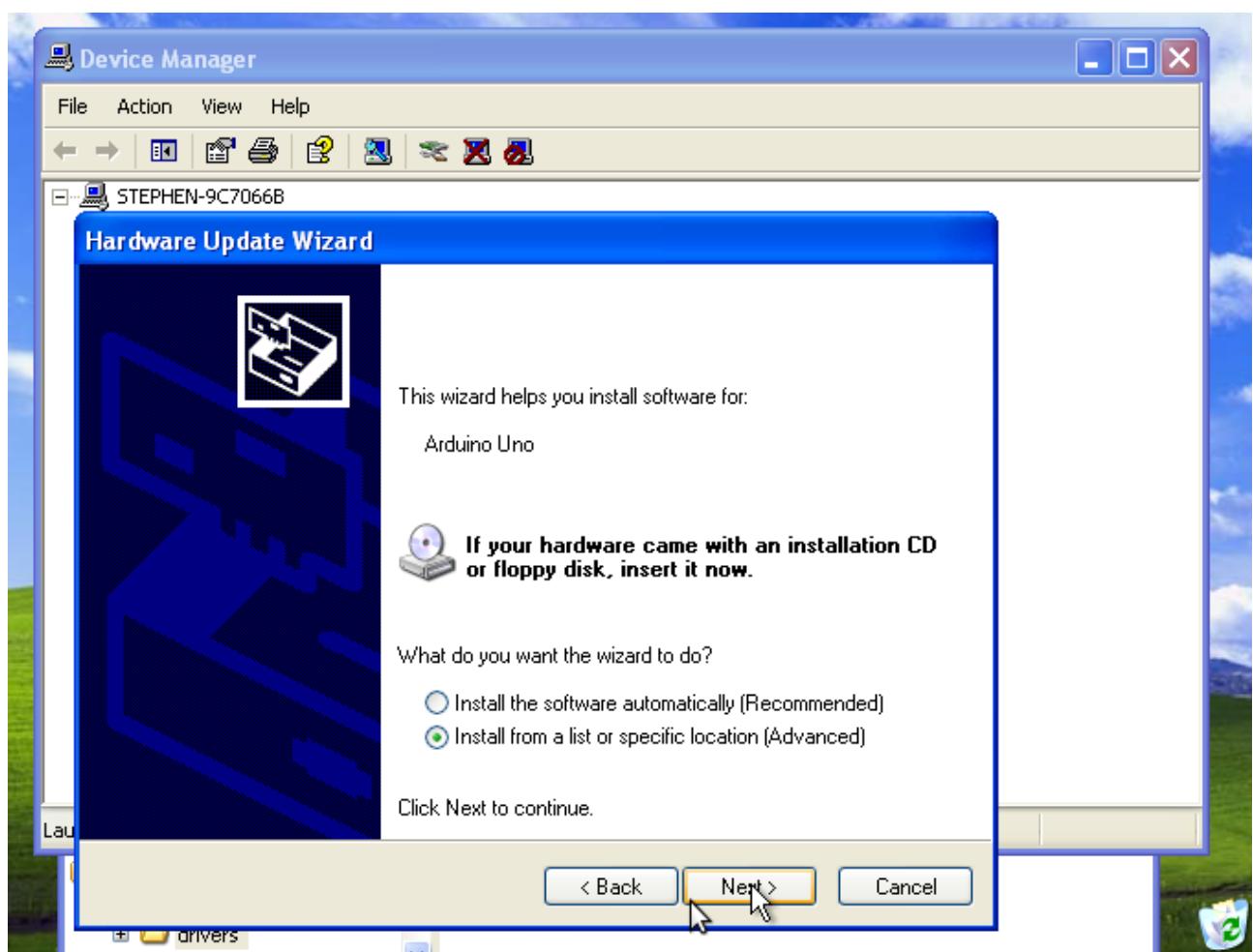
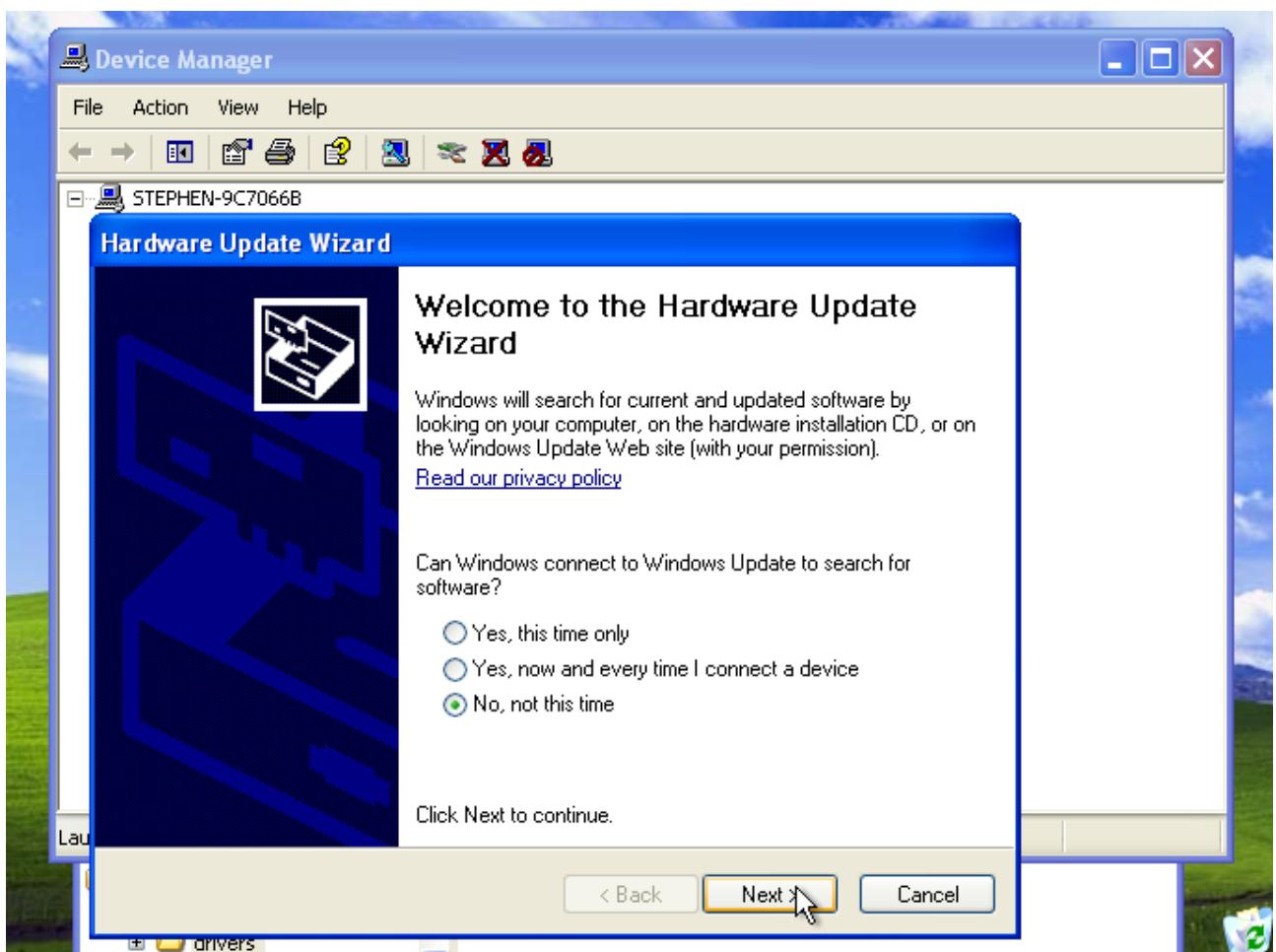


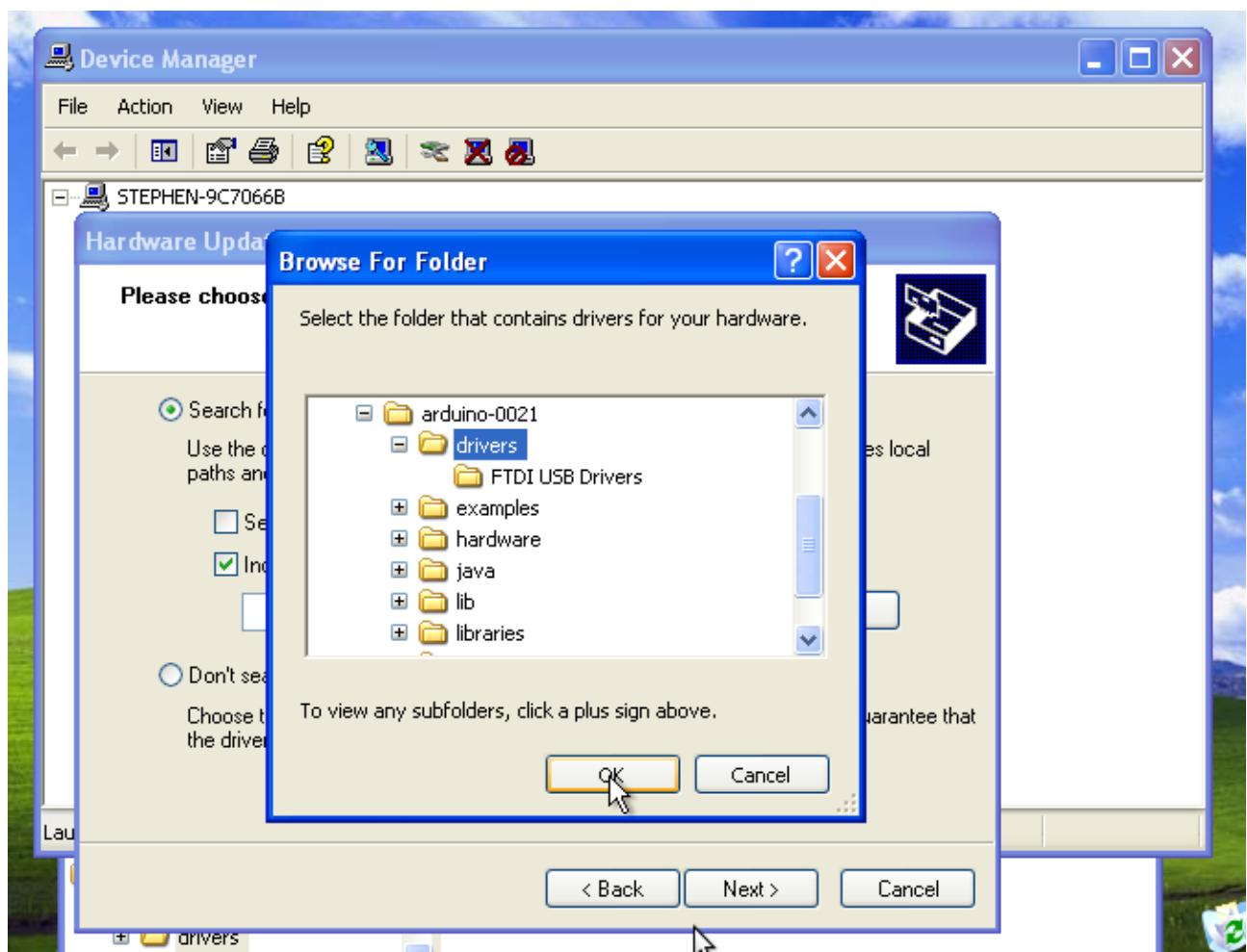
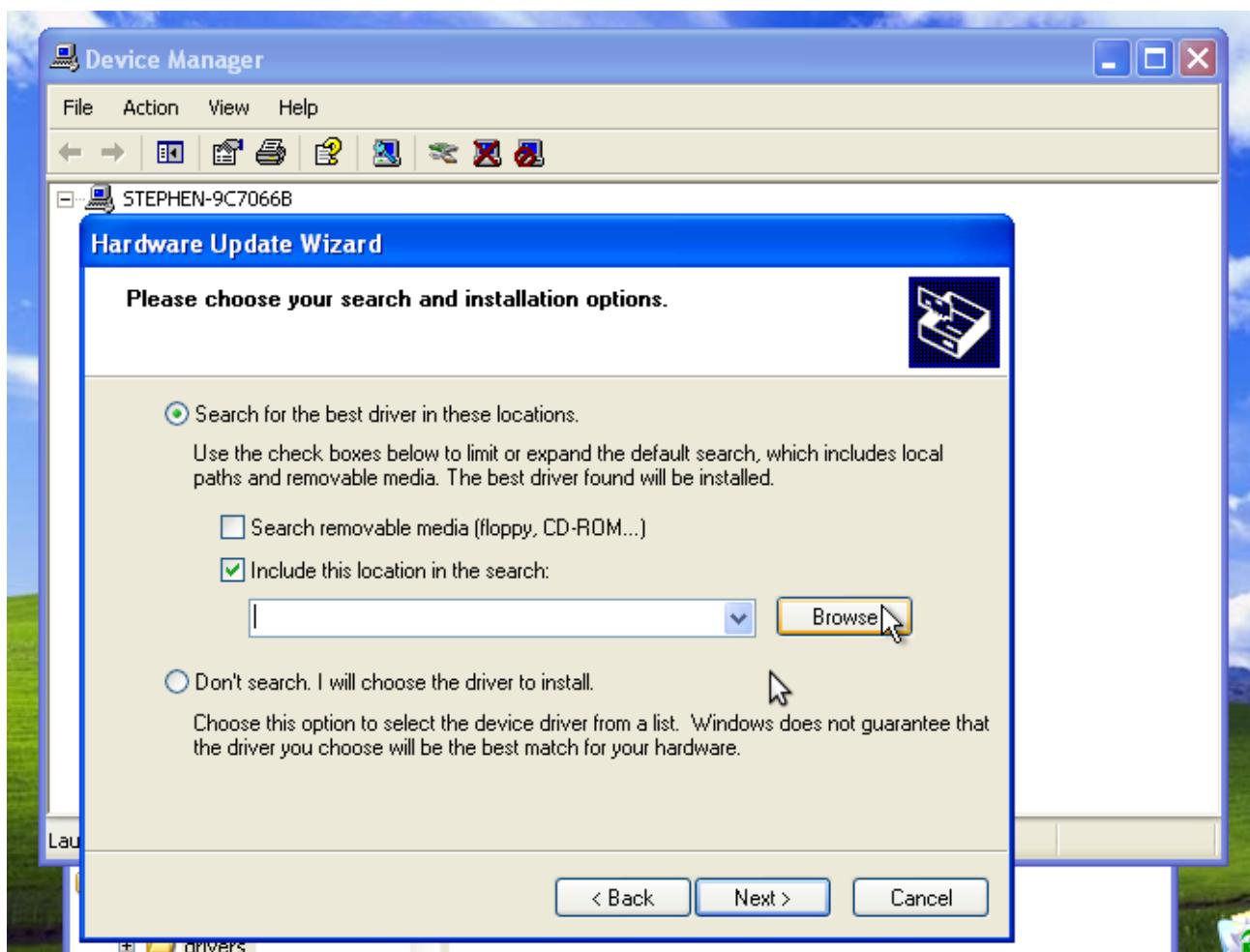
Arduino Windows Kurulumu

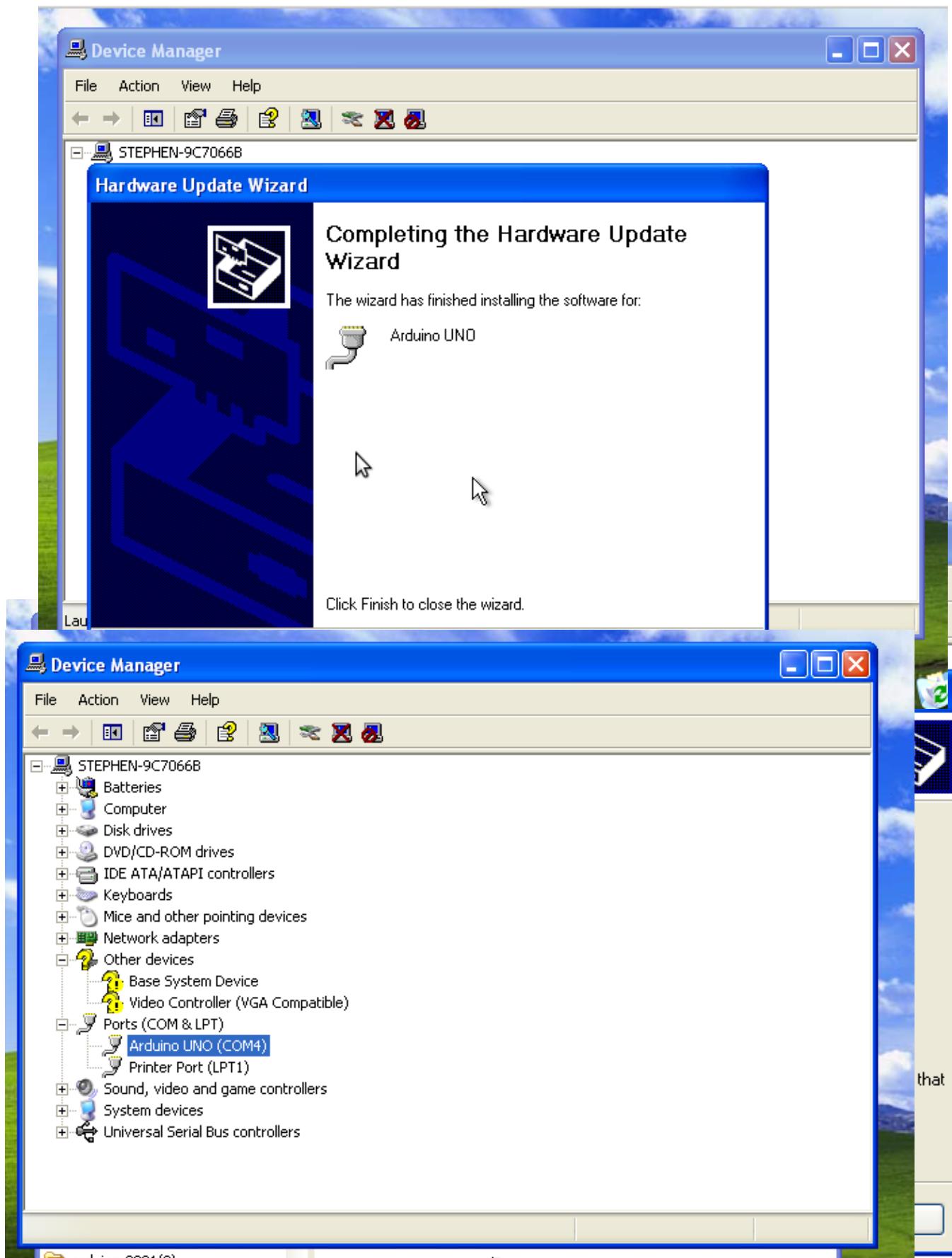
Öncelikle www.arduino.cc sitesinden arduinonun bilgisayarımızın işletim sistemine uygun IDE'yi indiriyoruz. İndirdikten sonra ise yapmamız gereken Arduino'yu bilgisayarımıza bağlayıp tanıtmak.

Bunu ekran görüntülerıyla anlatmak daha kolay sanırım. :)









Eger bu ekran görüntülerindeki adımları izlerseniz .Arduino'yu sorunsuz olarak bilgisayarınıza tanıtmışsınızdır.

Linux Üzerinde Arduino

Eger sizde benim gibi bir Ubuntu kullanıcısısanız Ubuntu Software Center'dan direk olarak kurabilirsiniz.Ya da Arduino'nun sitesinden linux için olan versiyonunu indirip direk olarak çalıştırabilirsiniz .Benim kişisel önerim ise bir linux dağıtıımı kullanmanızdır.Bir neden göstermek gerekirse 64 bit Windows üzerinde Processing(daha ilerde bahsedeceğim) seri monitor'u kullanamamanızdır.

PROGRAMLAMA

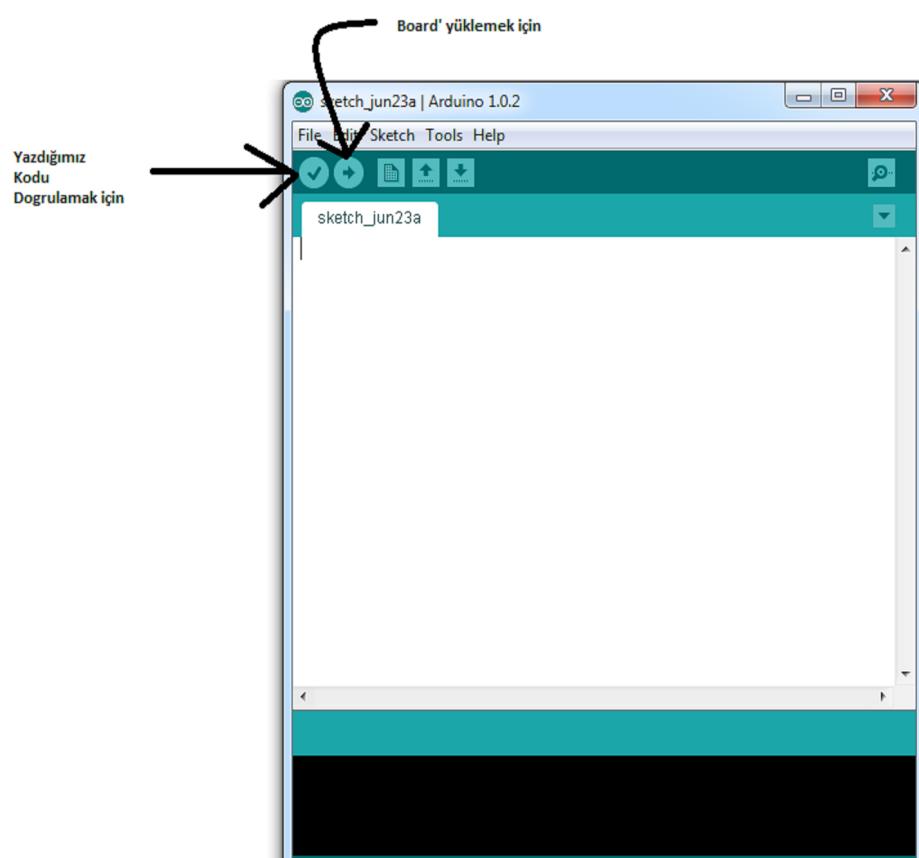
Arduino kod yapısı

Eger bir programlama diliyle uğraştıysanız Arduino size oldukça kolay gelecektir. Hiç ugraşmadıysanız ise korkmanıza gerek yok.Arduino'yu en temel seviyeden

anlatacağım. Bu sayede programlamaya ilginizi artırıp yazdığınız kodları çalıştırıp gözünüzle gördüğünüz için normal bir bilgisayar programlama dilinden daha kolay öğreneceksiniz.

Eğer C ailesinden bir dili ile ugraştıysanız Arduino dilini anlamanız daha kolay olacaktır. çünkü Arduino dili C dilinden esinlenmiştir. Ve C++ ile kütüphane yazabilir bunları kullanabilirsiniz.

Minimum Kod



Fazla uzatmadan artık programlamaya başlamak istiyorum.dedigim gibi Arduino oldukça kolay en başta bunu aklımızda tutalım.Zaten felsefesi kolay olmak. :)

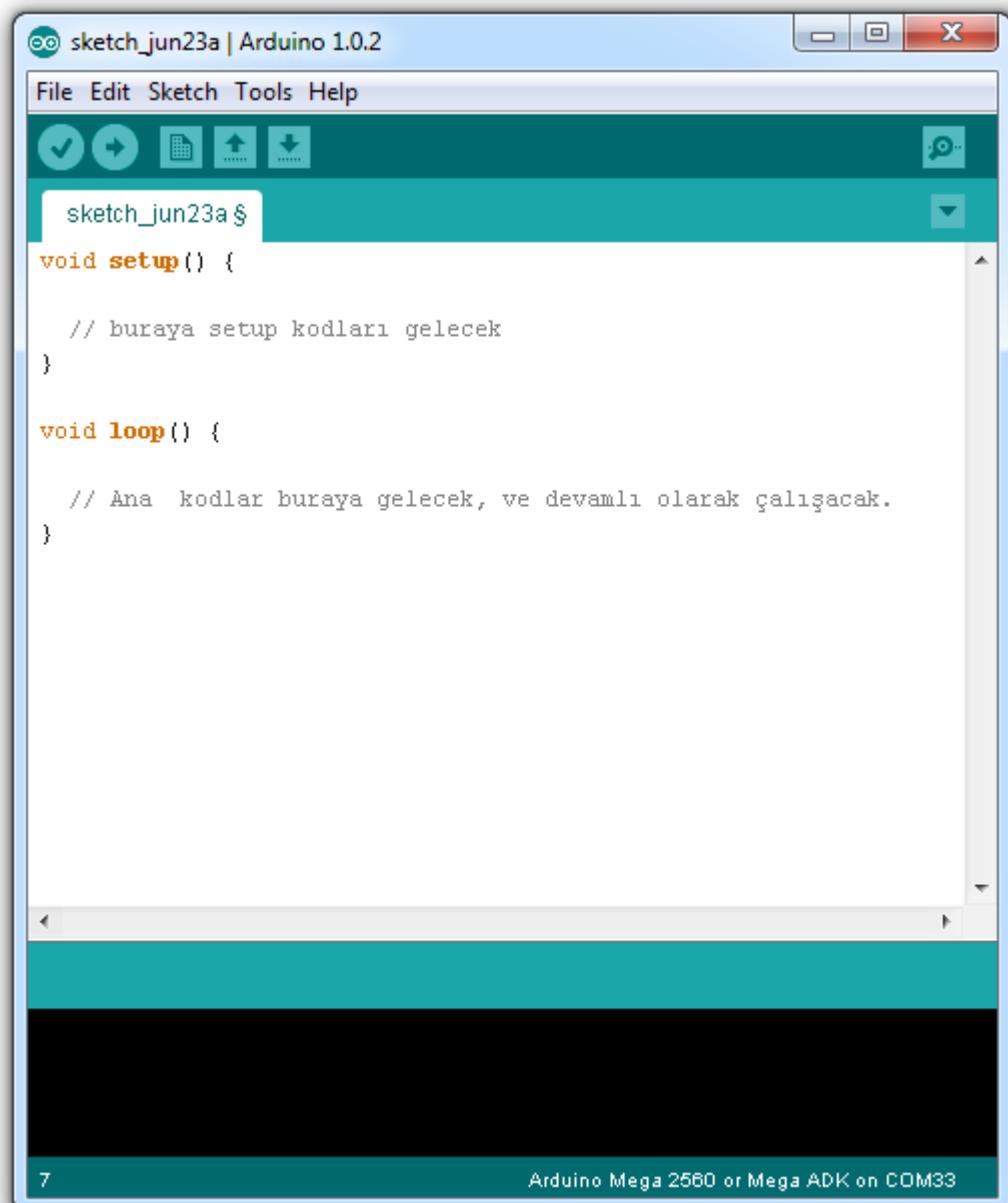
Arduino kodunun çalışabilmesi için minimum iki tane methoda ihtiyacımız var bunlar **setup()** ve **loop()** methodlarıdır.

Setup()

Bu fonksiyon sketch'in(her bir Arduino çalışmasına verilen ad) başladığını gösterir. Bu fonksiyonun içinde değişkenler ve pinler tanımlanır.Kütüphaneler kullanılmaya başlanır. Bu fonksiyon Arduino çalıştığında veya resetlendiğinde bir defa çalışır.

Loop()

Bu fonksiyon ise **Setup()** fonksiyonunda tanımlanan G/C birimlerinin kontrolünü bizim yazdığımız koda göre yapar. Örneğin bir led'e ne kadar süre yanıp ne kadar yanmayacağı burda yapabiliyoruz .yada bir LDR yardımıyla ışığın şiddetini ölçebiliriz.**Loop()** fonksiyonu Arduino çalışmaya başladıkten sonra devamlı olarak çalışır.devamlı olarak başa döner ve döngüyü devam ettirir.

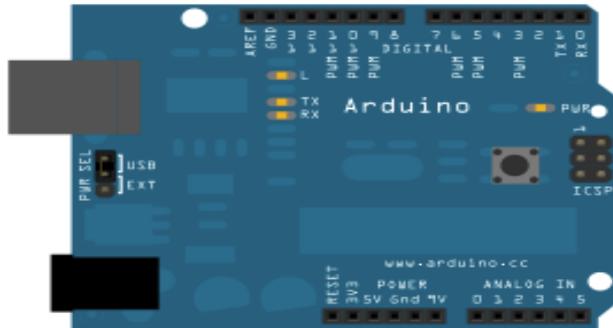


The screenshot shows the Arduino IDE interface with the title bar "sketch_jun23a | Arduino 1.0.2". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for file operations. The main window displays the following code:

```
void setup() {
    // buraya setup kodları gelecek
}

void loop() {
    // Ana kodlar buraya gelecek, ve devamlı olarak çalışacak.
}
```

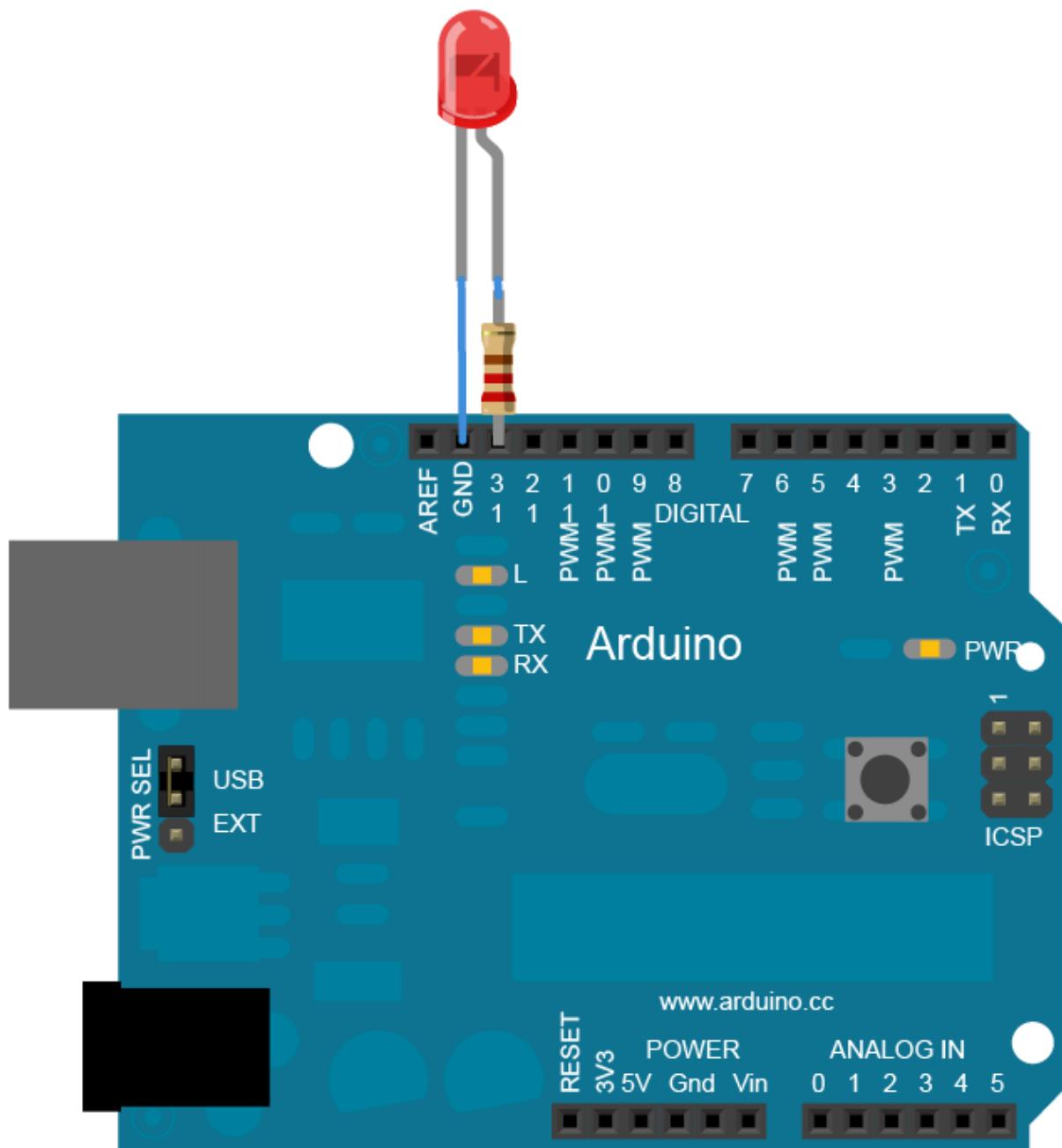
The code consists of two functions: **setup()** and **loop()**. The **setup()** function contains a comment indicating where setup code should be placed. The **loop()** function contains a comment indicating where the main program code should be placed, noting it will run continuously.



İlk Arduino Sketch'i

Buraya kadar Arduino nedir ne degildir diye bahsettim .Artık kodlamaya geçmenin zamanı geldi .
İlk kodumuzda bir Led'i yakıp söndürecegiz.

Devremizi şekildeki gibi kurduktan sonra tek yapmamız gereken yazdığımız kodu boarda yüklemek.



```
void setup() {
  // Burda dijital pini çıkış olarak başlatıyoruz.
  pinMode(led, OUTPUT);
}

//Loop'un içinde devamlı kodumuz devamlı dönüyor.
void loop() {
  // Burda LED'i yakıyoruz.
  digitalWrite(led, HIGH);

  delay(1000); // 1 saniye yakıyoruz .

  digitalWrite(led, LOW); // Burda ise LED'i söndürüyoruz.

  delay(1000); // 1 saniye söndürüyoruz.

}
```

Done Saving.

Eğer yukarıdakileri eksiksiz yaptıysanız şu an arduino

board'unuzda led'in yanıp söndüğünü görmelisiniz.

Eğer olmadıysa bu işlemlere tekrardan göz atmanızda fayda var. :)

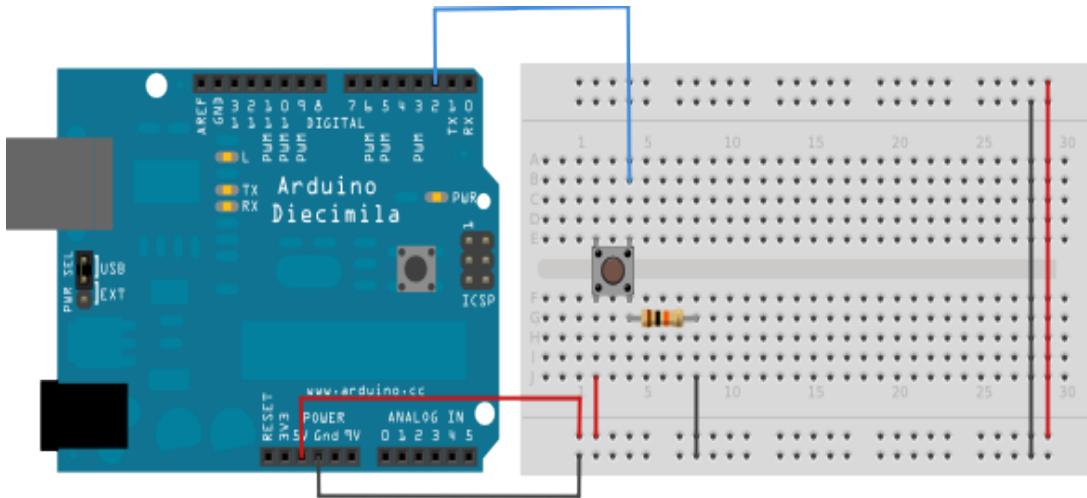
Seri Dijital Okuma

Bazı şeylerin Türkçesini tam bir çevirisi olmadığından ya da benim bilmediğimden dolayı böyle çeviriler olabilir.

Bu örnekte ise buton'un durumunu yani açık mı kapalımı onu Arduino board'umuzla seri iletişimini sağlayacağız.

Gerekli Donanımlar

- Arduino Board
- Buton
- 10k ohm direnç
- Devre Tahtası(Breadboard)
- Kablo



Yukardakı gibi butonu ve direnç bagladıkta sonra yapmamız gereken sadece kodumuzu board'a yüklemek.
Kodu File=>Examples=>Basics=>DigitalSerialRead yolunu izleyerek açabilirsiniz

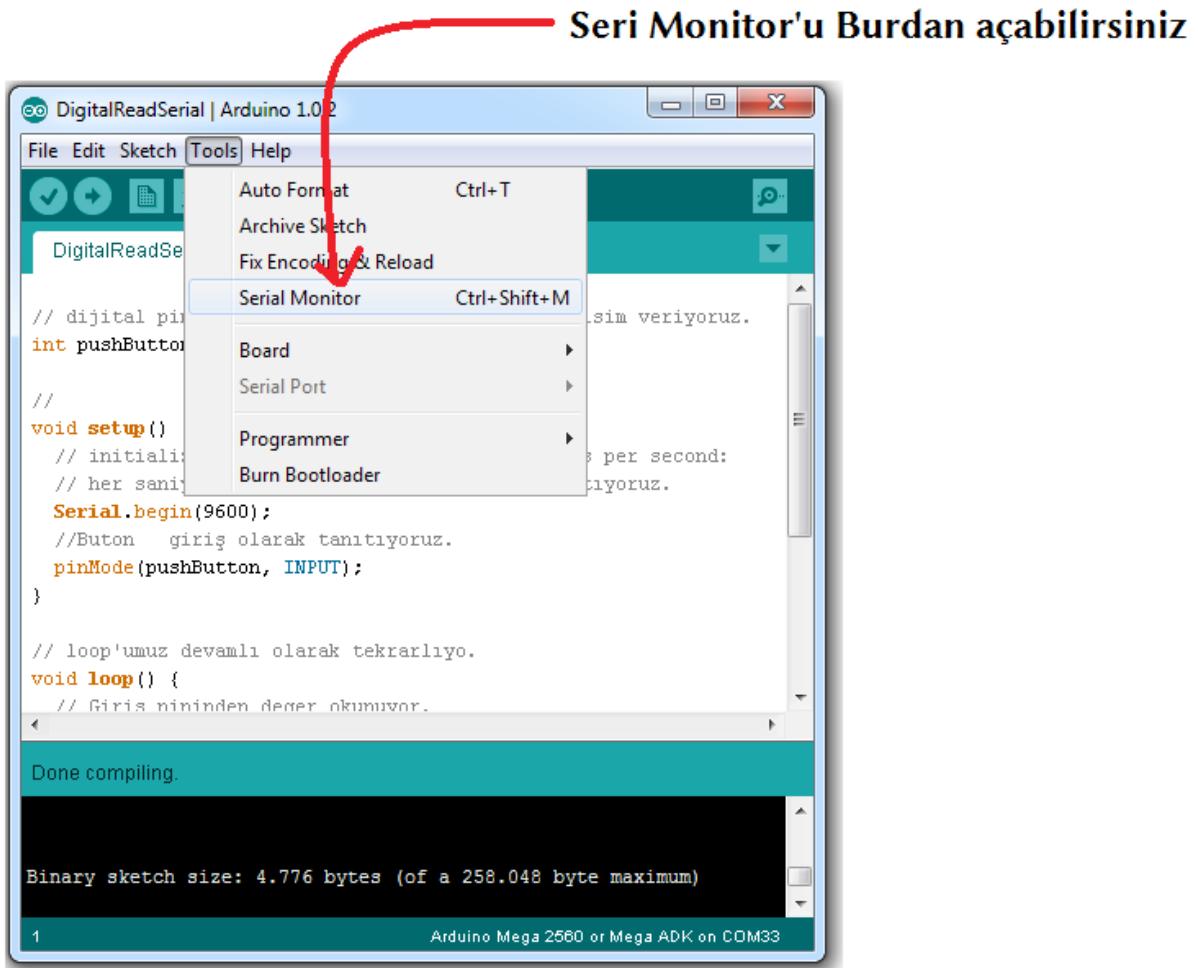
The screenshot shows the Arduino IDE interface with the title bar "DigitalReadSerial | Arduino 1.0.2". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for save, upload, and search. The main window displays the code for "DigitalReadSerial". The code initializes digital pin 2 for reading, sets up serial communication at 9600 bits per second, and defines the setup and loop functions. The setup function sets the pin mode to INPUT and begins serial communication. The loop function reads the digital input and prints it to the Serial Monitor. The status bar at the bottom indicates "1" and "Arduino Mega 2560 or Mega ADK on COM33".

```
// digital pin 2'ine butonu bağlıyoruz. ve bir isim veriyoruz.  
int pushButton = 2;  
  
//  
void setup() {  
    // initialize serial communication at 9600 bits per second:  
    // her saniyede 9600 bit seri iletişimini başlatıyoruz.  
    Serial.begin(9600);  
    //Buton giriş olarak tanıtiyoruz.  
    pinMode(pushButton, INPUT);  
}  
  
// loop'umuz devamlı olarak tekrarlıyo.  
void loop() {  
    // Giriş nininden değer okunuyor.  
<
```

Daha sonra yapmanız gereken Seri Monitor'u açıp dijital 1 ve 0 'ları görecegiz. Eğer devre açıksa 0 ,değilse 1 'i görecegiz.

Seri Monitor nerede diyorsanız aşağıdaki resimde

görebilirsiniz.

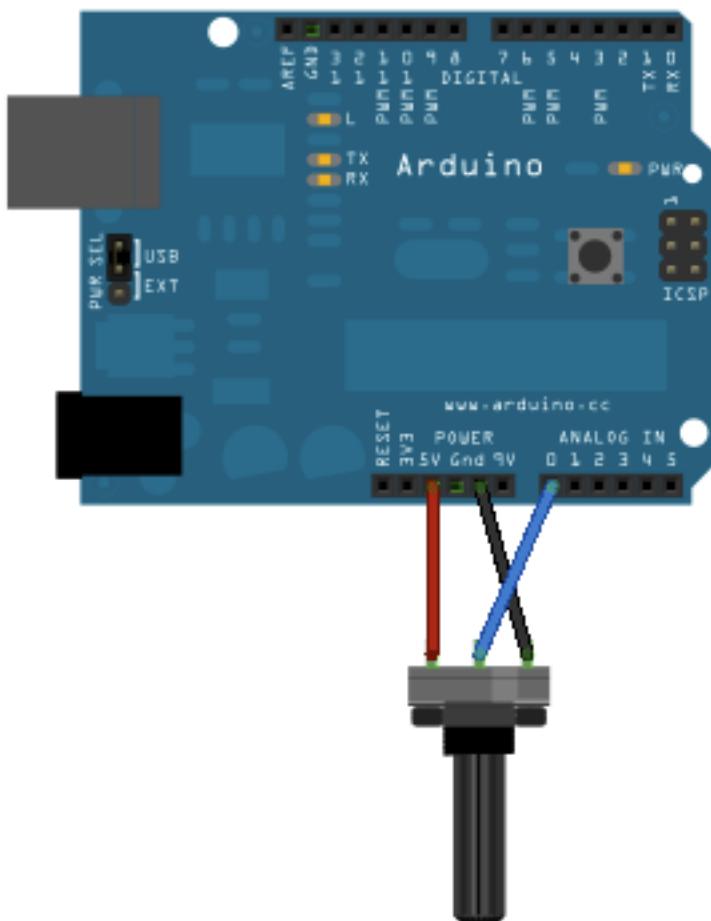


Analog Seri Okuma

Bu örnekte ise potansiyometre ile fiziksel dünyadan analog veriler okuyacağz. Öncelikle potansiyometrenin ne olduğundan biraz bahsedelim. Potansiyometre basitçe bir eksen üzerinde dönen ayarlı dirençtir.

Gerekli Donanımlar

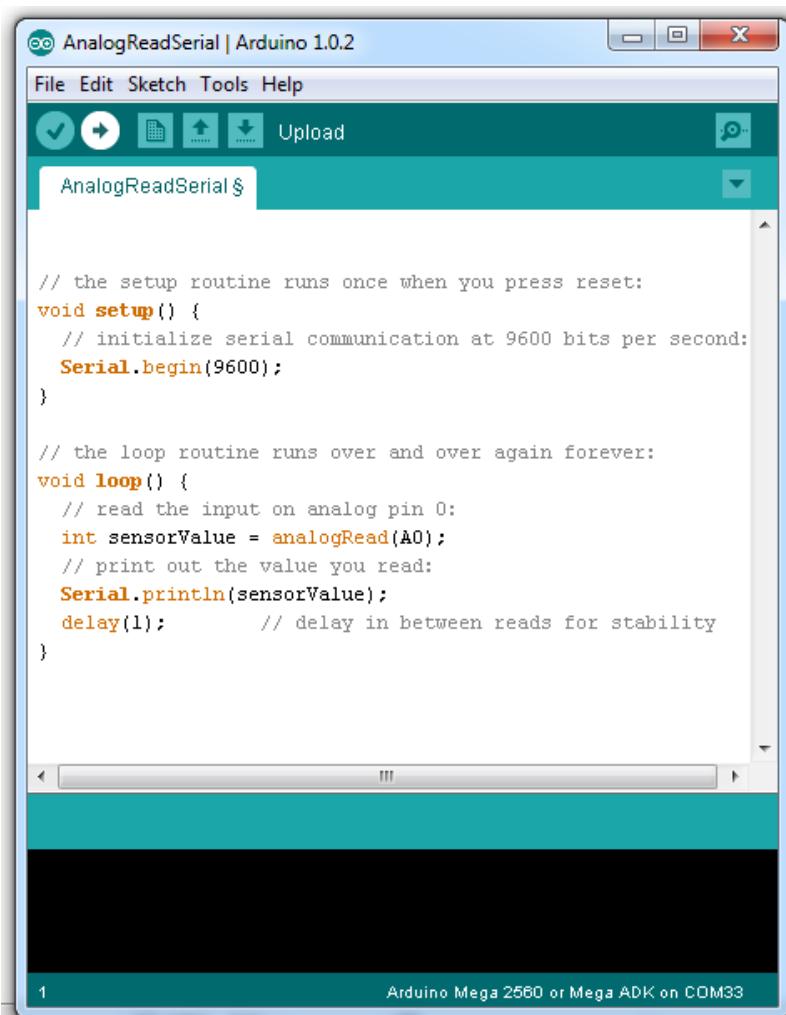
- Arduino
- 10 Kohm Potansiyometre



şekildeki gibi bağladıkten sonra tek yapmanız gereken Board'da kodu yüklemek.

Kodu File=>Examples=>Basics=>AnalogSerialRead yolunu izleyerek açabilirsiniz.

Kodu yükledikten sonra Seri Monitor'u açıp neler döndüğüne bakabilirsiniz. Eğer potensiyometreyle oynarsanız değerlerin değiştiğini göreceksiniz. Arduino'da bulunan ADC (analog to digital Converter) sayesinde 0-1023 arasında sayılara dönüştüğünü görebilirsiniz.



The screenshot shows the Arduino IDE interface with the title bar "AnalogReadSerial | Arduino 1.0.2". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, refresh, and other functions. The sketch code is displayed in the main editor area:

```
// the setup routine runs once when you press reset:  
void setup() {  
  // initialize serial communication at 9600 bits per second:  
  Serial.begin(9600);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
  // read the input on analog pin 0:  
  int sensorValue = analogRead(A0);  
  // print out the value you read:  
  Serial.println(sensorValue);  
  delay(1);          // delay in between reads for stability  
}
```

The bottom half of the window shows the Serial Monitor with a black background. The number "1" is visible in the top left corner of the monitor area, and the status bar at the bottom indicates "Arduino Mega 2560 or Mega ADK on COM33".

Buraya kadar basitçe analog ve digital kavramlarını anladığınızı varsayıarak artık programlamaya başlamak

istiyorum.

Sabitler ve Veri Tipleri

Sabitler Arduino dilinde öntanımlı olarak bulunmaktadır. Bu sabitler programımızın daha kolay okunmasını sağlar.

Boolean Sabitleri

Eğer az çok dijital elektronikle uğraştıysanız Boolean Matematiğinden haberdar olmalısınız.

Boolean sabitleri 1 ve 0 'dır .Arduino'da ise **true** ve **false** 'dir. Burda 0 **false** 1 ise **true** anlamına gelmektedir.

Pinlerin Durumlarını Belirleme

Arduino pinlerinden okuyabileceğimiz yada yazabileceğimiz iki tane değer vardır. Bunlar **HIGH** ve **LOW** 'dur.Burda HIGH voltajın var olduğunu LOW ise olmadığını gösterir.

digitalWrite(pin,HIGH) veya **digitalWrite(pin,LOW)** şeklinde kullanabiliriz.

Veri Tipleri

- **void**

Void kelimesi sadece fonksiyon tanımlanırken kullanılır.

Bu fonksiyon çağrıldığında hiçbir değer döndürmez.

Örnegin

```
void setup()
```

```
{
```

```
// ...
```

```
}
```

```
void loop()
```

```
{
```

```
// ...
```

```
}
```

- boolean

boolean'dan daha önce bahsettiğim için üzerinde durmayacağım .

- char

1 bayt'lık bir karakter değeri saklayan veri türüdür.

Örnek

```
char karakter='A';
```

Eğer birden çok karakter için

```
char karakter=" ABC" ;
```

şeklinde kullanılabilir.

Char (Karakterler) sayı olarak tutulabilir.ASCII tablosuna göz atmanız lazım.

Örnek

```
char karakter='A';
```

```
char karakter=65;
```

Eğer ASCII tablosuna göz attıysanız A'nın sayısal değerinin 65 olduğunu görebilirsiniz.

- **unsigned char**

Unsigned Char hafızada 1 baytlık yer tutar.**byte** ile aynıdır.

0-255 arasındaki sayıları encode eder.

- **Byte**

Byte , 0-255 arasında hafızada 8 bit unsigned sayı tutar.

- **int**

int veri tipi sayı tutmak için kullanılan birincil veri tipidir.
-32,768(- 2^{15})-32,767($2^{15}-1$) aralığında değer
alır.Hafızada 2 baytlık yer tutar.

Örnek

```
int ledPin=13;
```

- **unsigned int**

unsigned int ile int farkı ise unsigned int negatif değer
tutmaz.sadece pozitif değer alır. 0
-65,535($2^{16}-1$) arasındadır.int gibi hafızada 2 baytlık
yer tutar.

- **Word**

word ,0-65535 arasında 16 bit unsigned sayı
tutar.Unsigned int ile aynıdır.

- **Long**

Long,sayı tutmak için genişletilmiş veri tipidir.
-2,147,1483,648-2,147,483,647 arasında 32 bit'lik (4
bayts) sayı tutar.

- **unsigned long**

Unsigned long'ta long veri tipi gibi 32 bit (4 bayt) veri
tutar. Farkı ise sadece pozitif değerler almasıdır.
0-4,294,967,295($2^{32}-1$) aralığındadır.

- **short**

Short 16 bit'lik bir veri tipidir.

-32,768(- 2^{15}) -32,767($2^{15}-1$) arasındadır.

- **Float**

Float, decimal sayılar için kullanılan veri tipidir. Float veri tipi daha çok analog ve devam eden değerlerde daha büyük çözünürlük elde etmek için kullanılır. Değer aralığı -3,4028235E+38 ile 3,4028235E+38 arasındadır. 32 bit (4 bayt) sayı tutar. Float virgulden sonra 6-7 basamağa sahiptir.

- **Double**

Float'la aynı işi görür.

- **array(dizi)**

Dizi ,bir dizin numarası ile erişilen değişkenlerin topluluğudur.

Bir Dizi(Array) oluşturmak

```
int Dizi=[6];
```

```
int Pinler [ ]={2,4,8,7,6};
```

```
char mesaj[6] = " hello" ;
```

Dizinin bir elemanına değer atamak

```
mySensVals[0]=10;
```

Dizinin bir elemanından değer almak

```
x=mySensVals[4];
```

Dizi ve For Döngüsü

```
int i;  
for(i=0; i<5; i++)  
{  
    Serial.println(myPins[i]);  
}
```

OPERATÖRLER

Aritmetik Operatörler

- = atama operatörü

- + toplama operatörü
- - çıkarma operatörü
- * çarpma operatörü
- / bölme operatörü
- % mod(artık bölme)

Karşılaştırma Operatörleri

- == eşittir operatörü
- != eşit değil operatörü
- < küçüktür operatörü
- > büyüktür operatörü
- <= küçük veya eşittir
- >= büyük veya eşittir

Boolean Operatörleri

- **&& AND (ve) operatörü**

if deyiminde kullanılabilir.

Örnek

```
if (digitalRead(2) == HIGH && digitalRead(3) == HIGH)
{
    // read two switches
    // ...
}
```

- **|| OR (veya) operatörü**

```
if (x > 0 || y > 0) {
    // ...
}
```

- **! NOT (degil) operatörü**

```
if (!x) {
    // ...}
```

Bitwise Operatörleri

- & (bitwise and)
- | (bitwise or)
- ^ (bitwise xor)
- ~ (bitwise not)
- << (bitwise left)
- >> (bitwise right)

Birleştirme Operatörleri

- ++ artırma
- -- azaltma
- += ekleyerek atama
- -= eksilterek atama
- *= çarparak atama
- /= bölerek atama
- %= bölüm kalanı atama
- &=

- |=

Fonksiyonlar

Dijital G/Ç

- **pinMode()**

Bir giriş(input) ya da çıkış(output) olarak belirtilen pin'i yapılandırır.

Sözdizimi

pinMode(pin,Mode)

pin:Kaç numaralı pin'i ayarlamak istiyorsak onu yazıyoruz.

Mode: INPUT,OUTPUT veya INPUT_PULLUP durumlarından birini yazıyoruz.

Örnek

pinMode(led,OUTPUT);

- **DigitalWrite()**

HIGH veya LOW değerini pin'e gönderen fonksiyondur.

Eğer voltaj 5V veya 3.3V ayarlanmışsa HIGH değeri için bu değeri gönderir. LOW için ise 0V gönderir.

```
Blink | Arduino 1.0.5
File Edit Sketch Tools Help
Blink $ 

int led = 13;

// the setup routine runs once when you press reset:
void setup() {
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(led, HIGH);           // turn the LED on (HIGH is the voltage level)
    delay(1000);                    // wait for a second
    digitalWrite(led, LOW);          // turn the LED off by making the voltage LOW
    delay(1000);                    // wait for a second
}

Done compiling.

Binary sketch size: 1.632 bytes (of a 258.048 byte maximum)
```

Led' i yak

Led'i Söndür

• digitalRead()

Pin'in degerini okur .HIGH veya LOW olabilir.

deger = digitalRead(inPin);

```
sketch_aug11b | Arduino 1.0.5
File Edit Sketch Tools Help
sketch_aug11b §
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7; // pushbutton connected to digital pin 7
int val = 0; // variable to store the read value

void setup()
{
    pinMode(ledPin, OUTPUT); // sets the digital pin 13 as output
    pinMode(inPin, INPUT); // sets the digital pin 7 as input
}

void loop()
{
    val = digitalRead(inPin); // read the input pin
    digitalWrite(ledPin, val); // sets the LED to the button's value
}
```

Val degerini okur

Val degerini led'e yazar

16

Arduino Mega 2560 or Mega ADK on COM33

Analog G/Ç

- **analogReference()**

Analog giriş olarak kullanılan referans gerilimini yapılandırır. Seçenekler ise şunlardır

- DEFAULT: varsayılan analog 5V veya 3.3 Volt
- INTERNAL: 1.1 volt Atmega168 veya ATMEGA328 de 2.56 Volttur.
- EXTERNAL: AREF pin uygulanan voltajdır. 0-5V arasında kullanılır.

- **analogRead()**

Belirtilen analog pin değerini okur .Arduino 6 kanal (8 kanal Mini ve Nano, 16 kanal Mega) , 10 bit ADC(Analog Digital Converter). Bu şu anlama gelmektedir. 0 -1023 arasındaki tamsayı değerlere 0 ile 5 volta arası giriş gerilimleri ile eşleşir. $5\text{Volt}/1.024$ birim veya birim başına 0,0049. Yani Arduino'nun okuduğunuz Analog değeri bu sayı ile çarparak Voltajı bulabilirsiniz.

Bir analog giriş okumak için yaklaşık 100

mikrosaniye(0,0001) alır. En yüksek okuma oranı ise saniyede 10,000 defadır.

- **AnalogWrite**

Analog değeri belirtilen pin'e yazar.LED'lerin parlaklığını ayarlamak veya Motorları farklı hızlarda sürebilirsiniz.Kare dalga üretir.PWM sinyal frekansı yaklaşık olarak 490Hz'dır.

Çoğu Arduino Board'unda bu fonksiyon 3,5,6,9,10 ve 11 numaralı pinlerle çalışır.Sadece Mega board'ta 2 den 13 e kadardır.

Sözdizimi

analogWrite(pin,value)

Parametre

pin: pin'in numarası

value: 0(kapalı)-255(açık) arasındaki değerler

Örnek

Zaman

- **millis()**

Arduino'da yüklü mevcut olan programı başlamasından sonra milisaniye değerinde sayı döndürür.Yaklaşık 50

gün sonra overflow(taşma) olacaktır.

Örnek

- **micros()**

Arduino'da yüklü mevcut olan programı başlamasından sonra mikrosaniye değerinde sayı döndürür.Yaklaşık 70 dakika sonra overflow olacaktır.

Örnek

- **delay()**

Programı belirtilen süre kadar durdurur. Birimi milisaniyedir. 1000 milisaniye 1 saniyedir.

Sözdizimi

delay(ms)

Örnek

- **delayMicroseconds()**

Programı belirtilen süre kadar durdurur. Birimi mikrosaniyedir. 1000 mikrosaniye 1 milisaniyedir.Saniyenin milyonda bir'i kadardır.

Sözdizimi

delayMicroseconds(us)

Örnek

Matematik

- **min()**

iki sayıdan küçük olanı geri döndürür.

Örnek

SensValue=min(Sens,100) // burda devamlı küçük olan değeri geri döndürecektir. Yani 100 e kadar Sens değerini alacak 100 den sonra ise hep 100 değerini döndürecektir.

- **Max()**

iki sayıdan büyük olanı geri döndürür.

Örnek

SensValue=max(sens,20) // burda ise devamlı büyük olan değeri döndürecektir.Yani 20 ye kadar hep 20 sonrasında ise sens değerini döndürecektir.

- **Abs()**

sayının mutlak değerini geri döndürür.

- **Constrain()**

Sayıyı bir aralıkta sınırlar.

Örnek

`sensVal=constrain(sens,10,160);` // değişkenin değeri en küçük 10 en yüksek 160 olabilir.

- **Map()**

Map fonksiyonu sayının aralığını başka bir aralığa taşımaya yarar.

`y=map(x,1,50,50,1)` // burda birinci düşük değeri 1 iken 50 olmuş ikinci değeri ise 50 iken 1 olmuştur.

- **Pow()**

Sayının kuvvetini hesaplamaya yarayan fonksiyondur.

Sözdizimi

`pow(taban,üzeri)`

- **sqrt()**

Sayının karekökünü hesaplayan fonksiyondur.

Sözdizimi

`Sqrt(x)`

Trigonometri

- sin()
- cos()
- tan()

Rastgele(Random) Sayılar

- randomSeed()
- random

Bit'ler ve Bayt

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

İletişim

- Serial

Arduino board'nun bilgisayar veya diğer cihazlarla iletişimi için kullanılır. Bütün Arduinp board'larında en az bir tane Seri port bulunmaktadır. (Genellikle UART ve USART diye bilinirler). Seri port USB yardımı ile dijital pinler üzerindeki 0 (RX) ve 1 (TX) kullanarak haberleşir.

```
sketch_aug11a | Arduino 1.0.5
File Edit Sketch Tools Help

sketch_aug11a $

int incomingByte = 0; // for incoming serial data

void setup() {
    Serial.begin(9600); // opens serial port, sets data rate
}

void loop() {
    // send data only when you receive data:
    if (Serial.available() > 0) {
        // read the incoming byte:
        incomingByte = Serial.read();

        // say what you got:
        Serial.print("I received: ");
        Serial.println(incomingByte, DEC);
    }
}
```

Seri port açar ve hızını ayarlar

Seri verinin mevcut olup olmadığını kontrol eder

Seri veriyi ekranda görüntüler

Fonksiyonlar

- available()

Seri porttan okumak için kullanılabilir bayt sayısını alır.

Sözdizimi

Serial.available()

Örnek

- begin

Seri veri iletimi için saniyedeki bit (baud) hızını ayarlar.

300,600,1200,2400,4800,9600 ,

14400,19200,28800,38400,57600,115200 değerlerinden birini alır.

Sözdizimi

Serial.begin(hız)

- end()

Seri iletişim devre dışı bırakır.pinleri genel giriş ve çıkış olarak kullanmaya izin verir. Yeniden iletişime geçmek için Serial.begin() kullanılır.

Sözdizimi

Serial.end()

- Find()

Verilen uzunlukta veri bulunana kadar tampon verileri okur. Hedef dize, yanlış zaman veya olmadığı bulunursa işlevi true değerini döndürür.

Sözdizimi

Serial.find(hedef)

- findUntil()

Belirli bir uzunlukta veya sonlandırıcı bir dize bulunana kadar Serial.findUtil() tampon verilerini okur.

Sözdizimi

Serial.findUtil(hedef,terminal)

- Flush()

Giden Seri verinin tamamlanmasını bekler.

Sözdizimi

Serial.flush()

- parseFloat()

Seri verinin virgulden sonrasında geri döndürür. Basamak (veya eksik işaret) olmayan karakterler atlanır.

Sözdizimi

Serial.parseFloat()

- `parseInt()`

Seri veride bir sonraki geçerli tam sayıyı arar.

Sözdizimi

Serial.parseInt()

- `peek()`
- `print`

Seri port'tan aldığı verileri ASCII tablosundaki karşılığı şeklinde yansıtır. Bu komut birçok şekilde olabilir.

Örnek

- `Serial.print(78)` "78" verir
- `Serial.print(1.23456)` "1.23" verir
- `Serial.print ('N')` "N" verir.
- `Serial.print("Merhaba")` "Merhaba" verir.

Örnek

- `println()`

`print()` 'den farkı daha okunabilir veriyi göstermesidir. Veri bittiginde diğer veriyi bir alt satırda gösterir.

Sözdizimi

Serial.println(değer)

Serial.println(değer,format)

- Read()

Gelen seri veriyi okur.

Sözdizimi

Serial.read()

- readBytes
- readBytesUntil()
- setTimeout()
- write

Seri port'a ikilik verileri yazar.Bu veri bayt bayt veya dizi olarak gönderilir.

Sözdizimi

Serial.write(val)

Örnek

- Stream

Kontrol Yapıları

- **if ve if ...else**

if ve else koşullu işlem yapan deyimlerdir.if ve else tek bir karşılaştırma deyimi olup else kullanımı istege bağlıdır.Eğer koşul doğru ise if den sonraki bölüm yürütülür. Else'den sonraki bölüm atlanır.

Örnek-1

```
if(degisen>50)
{
    digitalWrite(ledPin,HIGH)
}
```

Örnek -2

```
if(degisen>500)
{
    digitalWrite(ledpin,LOW)
}
else
{
    digitalWrite(ledpin,HIGH)
}
```

- **for**

Bu deyim, diğer döngü deyimleri gibi bir kümeyi bir çok kez tekrarlamak için kullanılır. Koşul sınaması while da olduğu gibi döngüye girmeden yapılır. Bu döngü deyimin içinde diğerlerinden farklı olarak başlangıç değeri ve döngü sayacına sahip olmasıdır.

Örnek

```
int PWMpin = 10; // led digital 10 'a bağlanır.  
void setup()  
{  
    // no setup needed  
}  
void loop()  
{
```

```
for (int i=0; i <= 255; i++){  
    analogWrite(PWMpin, i);  
    delay(10);  
}  
}
```

- **switch case**

Bu deyim bir değişkenin içeriğine bakarak, programın akışını bir çok seçenekten birine yönlendirir. case (durum) deyiminden sonra değişkenin durumu belirlenir ve takip eden gelen satırlar (deyimler) işleme konur. Bütün durumların aksi söz konu olduğunda gerçekleştirilmesi istenen deyimler default deyiminden sonraki kısımda bildirilir.

Örnek

```
switch (degisken) {  
    case 1:  
        //değişken 1 eşit olduğunda yürüt  
        break;  
    case 2:  
        //değişken 2 eşit olduğunda yürüt
```

```
break;  
default:  
// Hiçbiri ile eşleşmez ise yürüt  
// default deyimi opsyoneldir.  
}
```

- **while**

Tekrarlama deyimidir. Bir küme ya da deyim while kullanılarak bir çok kez yinelenebilir. Yinelenmesi için koşul sınaması döngüye girilmeden yapılır. Koşul olumlu olduğu sürece çevrim yinelenir.

Örnek

```
degisken = 0;  
while(degisken < 200){  
// 200 defa kodu yürütür.  
degisken++;  
}
```

- **do....while**

Bu deyimin while deyiminden farkı, koşulun döngü sonunda sıyanmasıdır. Yani koşul sıyanmadan döngüye girilir ve döngü kümlesi en az bir kez yürütülür. Koşul olumsuz ise döngüden sonraki satıra geçilir.

Örnek

do

{

```
delay(50);      //sensor stabilizisyanu için bekle  
x = readSensors(); // Sensoru oku
```

} while (x < 100);

- **break**

Bir Arduino programında, bir işlem gerçekleştirilirken, işlemin sona erdirilmesi bu deyim ile yapılır. Örneğin, döngü deyimleri içindekiler yürütülürken, çevrimin, koşuldan bağımsız kesin olarak sonlanması gerekiğinde bu deyim kullanılır.

Örnek

```
for (x = 0; x < 255; x ++)
```

```
{  
  digitalWrite(PWMpin, x);  
  sens = analogRead(sensorPin);  
  if (sens > threshold){  
    x = 0;  
    break;  
  }  
  delay(50);  
}
```

- **continue**

Bir döngü içerisinde continue deyimi ile karşılaşılırsa, ondan sonra gelen deyimler atlanır ve döngü bir sonraki çevrime girer.

Örnek

```
for (x = 0; x < 255; x ++)  
{  
  if (x > 40 && x < 120){ // create jump in values  
    continue;  
}
```

```
digitalWrite(PWMpin, x);  
delay(50);  
}
```

- **return**

Fonksiyon'u sonlandırmak veya değer döndürmek için kullanılır.

Örnek

```
int checkSensor(){  
    if (analogRead(0) > 400) {  
        return 1;  
    }  
    else{  
        return 0;  
    }  
}
```

- **goto**

Programın herhangi bir yerinden başka bir yerine atlamak için goto deyi̇mi̇ kullanılır.

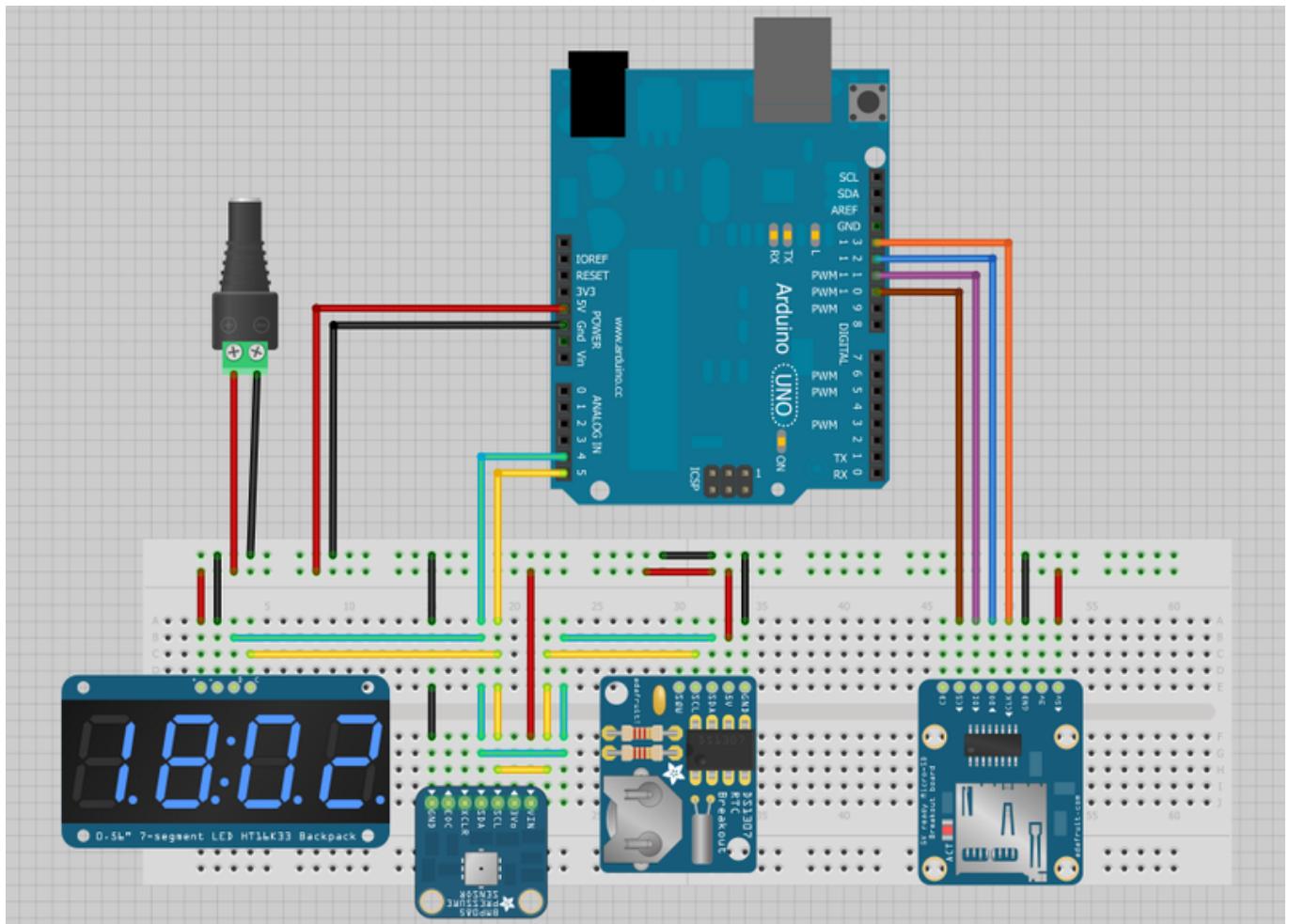
Örnek

```
for(byte r = 0; r < 255; r++){  
    for(byte g = 255; g > -1; g--){  
        for(byte b = 0; b < 255; b++){
```

```
if (analogRead(0) > 250){ goto bailout;}  
    // more statements ...  
}  
}  
}  
bailout:
```

FRITZING

Fritzing açık kaynak kodlu geliştirilen devre tasarım programıdır. Fritzing içinde bulunan Arduino board'larıyla (diger board'larda mevcut) ve temel devre elemanlarıyla devrenizi çabuk prototip üretmenizi sağlar. Fritzing'i kullanmak için çok fazla Bir şey bilmenize gerek yok devre elemanlarını sürükle bırak ile çalışma alanınıza alıp kullanabilirsiniz.



Fritzing www.fritzing.org adresinden işletim sisteminize uygun olan versiyonunu indirebilirsiniz. Ayrıca fritzing Windows ,Linux ve Mac OS X çalışmaktadır.

PROCESSING

Processing, resim, animasyon ve etkileşim yöntemleri geliştirebiliniz açık kaynak bir programlama dili ve ortamıdır. Çoklu ortam malzemeleriyle (resim, video ve ses gibi) sorunsuz çalışabilir, işleyebilir. Görsel bir çıktı üzerinden

hareket ederek bilgisayar programlamanın prensipleri ve temellerini disiplin dışı kişilerinde çok rahat ve hızlı bir şekilde öğrenmesine olanak sağlar. O nedenle birçok araştırmacı, öğrenci, tasarımcı ve sanatçı tarafından tercih edilen bir programlama dilidir. Kendi yaptıkları profesyonel işlerinde prototip uygulama, sergi ögesi veya işlerinin niteliklerini artıracak nitelikte kullanmaktadır. Processing bu kadar farklı amaçlarda farklı insanların tercih etme sebeplerini sıralayacak olursak;

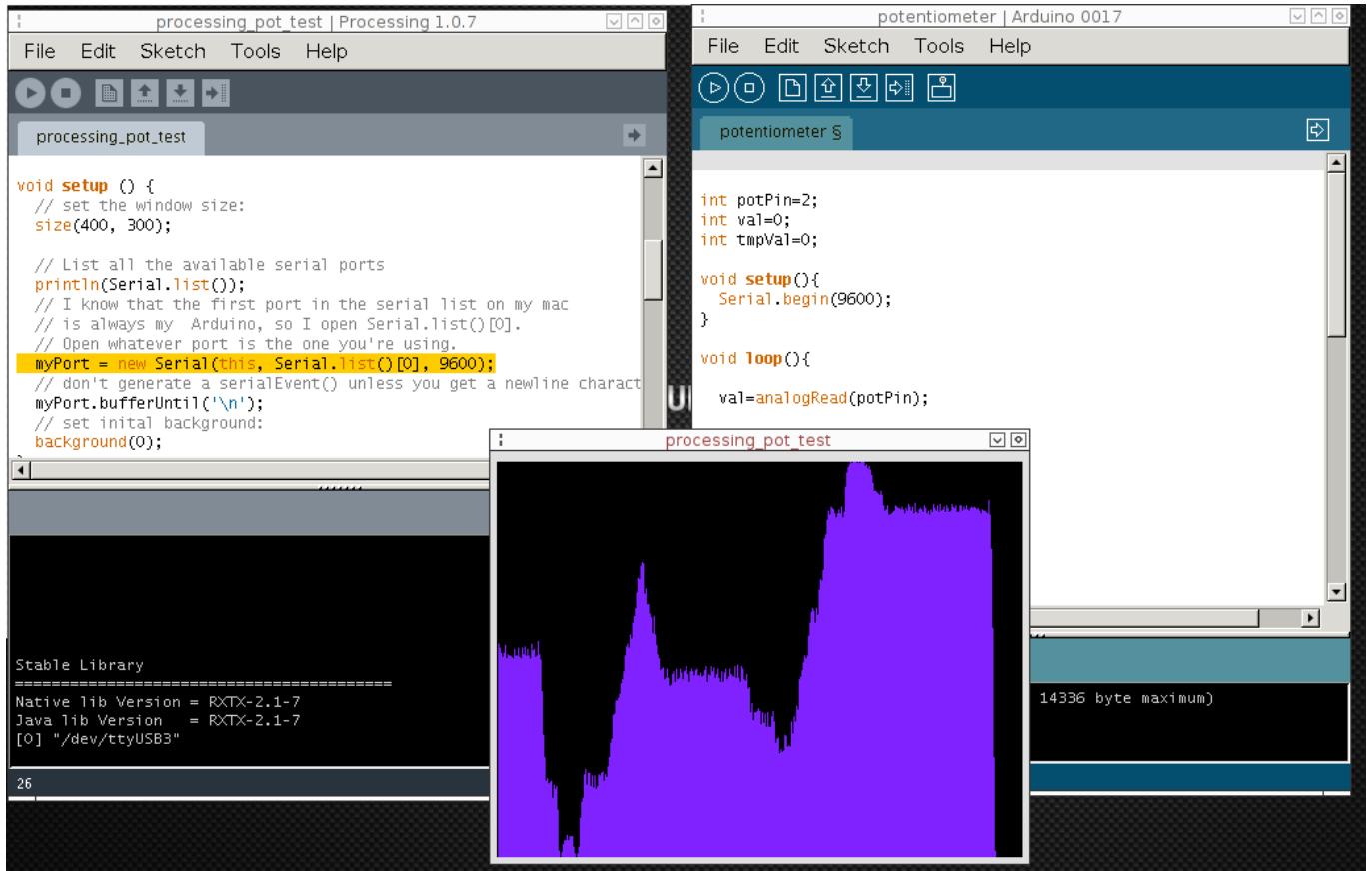
- Açık kaynak ve tüm işletim sistemleri ile sorunsuz çalışabilmesi
- 2D ve 3D ile sorunsuz çalışabilme bu görüntüler üzerinde kolay ve hızlı etkileşim yaratabilme (gelişmiş 2D ve 3D kütüphaneleri)
- Uygulama içersinden basit ve kolay PDF çıktılar alabilme (zengin PDF kütüphaneleri)
- OpenGL entegrasyonu ile 3D motor kullanımı
- Ses, video başta olmak üzere 100' den fazla kütüphane ile birçok çoklu ortam ögesi ile sorunsuz çalışabilme
- Web ve masaüstü' de çalışabilir uygulamalar yaratabilme
- Andorid desteği

- Sadece processing için özelleştirilmiş bir programlama ortamı (PDE – Processing Development Environment)
- Geniş kaynak erişimi (Güncel ve eksiksiz dokümantasyon ve özel gereksinimler üzerine yazılmış çokça sayıda kitap)

Processing'den bahsetmemin nedeni Arduino ile Seri veri iletimi sayesinde Data Visualization kolayca yapılabilmesidir. Processing ile zaten zevkli olan Arduino programlama daha zevkli hale geliyor. Birçok projeniz görsel anlamda daha göze hitap edecek hale geliyor ve insanların anlaması kolaylaşıyor.

Processing 'i <https://processing.org/download/> adresinden indirebilirsiniz. Kişisel bir tavsiye olarak eger 64 bit windows kullanıyorsanız Processing'in

(İlerde bir iki örnekle processing'i anlat)



PROJELER

• Proje-1

Aslında ilk projelerimizi yapmış olmakla birlikte burda ise sadece proje bazlı anlatacağım.

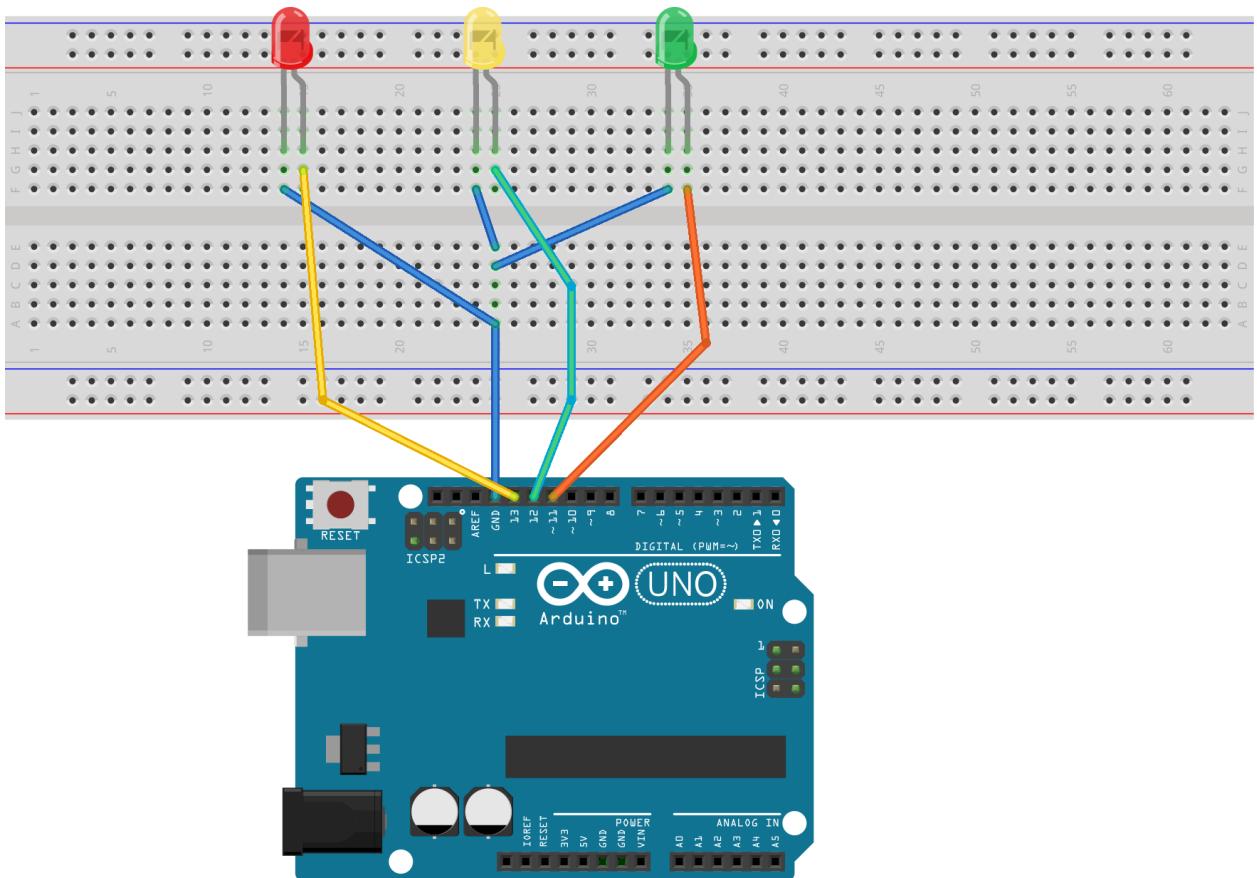
LED yakıp söndürdüğümüz için ilk projemiz biraz daha yine led'lerle alakalı olup biraz daha kullanışlı bir proje olacak

Trafik Lambası

Projemizin adından anlaşılacağı üzere trafik lambası yapacağız.

Neye ihtiyaçın var

1. BreadBoard
2. 1'er adet kırmızı ,sarı ve yeşil led
3. Jumper kablo



Made with Fritzing.org

Devremizi şekildeki gibi bağladıkta sonra tek yapmamız gereken kodu yüklememiz.

Kod

```
sketch_aug11a | Arduino 1.0.5
File Edit Sketch Tools Help
sketch_aug11a
int kirmizi=13;
int sari=12;
int yesil=11;

void setup()
{
    pinMode(kirmizi,OUTPUT);
    pinMode(sari,OUTPUT);
    pinMode(yesil,OUTPUT);
}

void loop()
{
    digitalWrite(kirmizi,HIGH);
    digitalWrite(sari,LOW);
    digitalWrite(yesil,LOW);
    delay(30000);
    digitalWrite(kirmizi,LOW);
    digitalWrite(sari,HIGH);
    digitalWrite(yesil,LOW);
    delay(6000);
    digitalWrite(kirmizi,LOW);
    digitalWrite(sari,LOW);
    digitalWrite(yesil,HIGH);
    delay(30000);
}
```

pin'leri burda belirliyoruz

Led'ler bir çıkış olduğundan OUTPUT değerini yazıyoruz

Bu Blok ise programınızın ana akışını oluşturmaktadır. Ve kodun devamlı olarak çalışmasını sağlayacaktır.

pin'leri burda belirliyoruz

Led'ler bir çıkış olduğundan OUTPUT değerini yazıyoruz

Bu Blok ise programınızın ana akışını oluşturmaktadır. Ve kodun devamlı olarak çalışmasını sağlayacaktır.

Kırmızı LED açık Sarı ve Yeşil Kapalı

30 saniye kırmızının açık diğerlerinin kapalı olmasını sağlayacak süre

Sarı açık diğerleri kapalı

6 saniye boyunca Sarı açık

Yeşि açık diğerleri kapalı

30 saniye boyunca yeşil açık

Done compiling.

Binary sketch size: 1.718 bytes (of a 258.048 byte maximum)

2 Arduino Mega 2560 or Mega ADK on COM33

