ARDUİNO NEDİR?

Arduino, kullanımı kolay donanım ve yazılıma dayalı açık kaynak kodlu bir elektronik platformudur. Arduino kartları, girişleri okuyabilir - bir sensördeki ışığı, bir düğmedeki parmğı veya bir Twitter mesajını - ve bunları bir çıktıya dönüştürebilir - bir motoru etkinleştirme, bir LED'i yakma, çevrimiçi bir şeyler yayınlama gibi. Kart üzerindeki mikrodenetleyiciye bir dizi talimat göndererek kartınıza ne yapacağını söyleyebilirsiniz. Bunu yapmak için Arduino programlama dilini (Wiring'e dayalı) ve Arduino Yazılımını (IDE) Processing'e dayalı kullanırsınız.

Yıllar boyunca Arduino, günlük nesnelerden karmaşık bilimsel aletlere kadar binlerce projenin beyni olmuştur. Öğrenciler, hobiciler, sanatçılar, programcılar ve profesyonellerden oluşan dünya çapında bir üretici topluluğu bu açık kaynaklı platform etrafında toplanmıştır.

Arduino, elektronik ve programlama konusunda geçmişi olmayan öğrencilere yönelik, hızlı prototipleme için kolay bir araç olarak Ivrea Etkileşim Tasarım Enstitüsü'nde doğdu. Daha geniş bir topluluğa ulaşır ulaşmaz, Arduino kartı yeni ihtiyaçlara ve zorluklara uyum sağlamak için değişmeye başladı ve basit 8 bitlik kartlardan IoT uygulamalarına, giyilebilir 3D baskı ve gömülü ortamlar için ürünlere kadar farklılaştı.

Neden Arduino?

Arduino, basit ve erişilebilir kullanıcı deneyimi sayesinde binlerce farklı proje ve uygulamada kullanılmıştır. Arduino yazılımı yeni başlayanlar için kullanımı kolay, ancak ileri düzey kullanıcılar için de yeterince esnektir. Mac, Windows ve Linux'ta çalışır. Öğretmenler ve öğrenciler düşük maliyetli bilimsel aletler yapmak, kimya ve fizik prensiplerini kanıtlamak veya programlama ve robotiğe başlamak için kullanırlar. Arduino, yeni şeyler öğrenmek için önemli bir araçtır. Herkes - çocuklar, hobiciler, sanatçılar, programcılar - sadece bir kitin adım adım talimatlarını izleyerek veya Arduino topluluğunun diğer üyeleriyle çevrimiçi fikir paylaşarak kurcalamaya başlayabilirler.

Fiziksel hesaplama için birçok başka mikrodenetleyici ve mikrodenetleyici platformu mevcuttur. Arduino bunların mikrodenetleyici programlamanın karmaşık ayrıntılarını alır ve bunları kullanımı kolay bir pakette bir araya getirir. Arduino ayrıca mikrodenetleyicilerle çalışma sürecini basitleştirir, ancak öğretmenler, öğrenciler ve ilgili amatörler için diğer sistemlere göre bazı avantajlar sunar:

- **Ucuz** - Arduino kartları diğer mikrodenetleyici platformlarına kıyasla nispeten ucuzdur. Arduino modülünün en ucuz versiyonu elle monte edilebilir ve önceden monte edilmiş Arduino modülleri bile \$50'den daha ucuza mal olur
- **Platformlar arası** - Arduino Yazılımı (IDE) Windows, Macintosh OSX ve Linux işletim sistemlerinde çalışır. Çoğu mikrodenetleyici sistemi Windows ile sınırlıdır.
- **Basit, net programlama ortamı** - Arduino Yazılımı (IDE) yeni başlayanlar için kullanımı kolay, ancak ileri düzey kullanıcıların da faydalanabileceği kadar esnektir.

Avrupa Birliği tarafından ortak finanse edilmektedir

- **Açık kaynak ve genişletilebilir yazılım** - Arduino yazılımı, deneyimli programcılar tarafından genişletilebilen açık kaynak araçları olarak yayınlanmıştır. Dil, C++ kütüphaneleri aracılığıyla genişletilebilir ve teknik ayrıntıları anlamak isteyen kişiler Arduino'dan, temel aldığı AVR C programlama diline geçiş yapabilir. Benzer şekilde, isterseniz AVR-C kodunu doğrudan Arduino programlarınıza ekleyebilirsiniz.

ARDUİNO'YA BAŞLARKEN

Donanım, yazılım araçları ve Arduino API'sine giriş.

Arduino platformu, 2005 yılındaki başlangıcından bu yana elektronik ve gömülü tasarım alanında en tanınmış markalardan biri haline geldi.

Peki Arduino'nun temel taşları nelerdir? "Kart" nedir, ona nasıl kod yazarım ve kendi projemi oluşturmak için gereken araçlar nelerdir? Bu kılavuzun amacı size Arduino projesine genel bir bakış sağlamaktır.

Arduino Donanımı

Arduino, yıllar boyunca birçok şekil ve formda yüzlerce donanım tasarımı yayınladı. Bunlar;

Klasik Arduino ailesi

			
Arduino UNO R4 Minimum	Arduino UNO R4 Wi-Fi	Arduino UNO R3	Arduino Leonardo
			
Arduino UNO Mini Sınırlı Sayıda Üretim	Arduino Mikro	Arduino Sıfır	Arduino UNO WiFi Rev2

Nano Ailesi

Nano Ailesi, özelliklerle dolu, küçük bir ayak izine sahip bir dizi karttır. Bu kartlarda sıcaklık/nem, basınç, hareket, mikrofon ve daha fazlası gibi bir dizi gömülü sensör bulunur. Ayrıca MicroPython ile programlanabilir ve Makine Öğrenmesini destekler.

Avrupa Birliği tarafından ortak finanse edilmektedir

MKR Ailesi

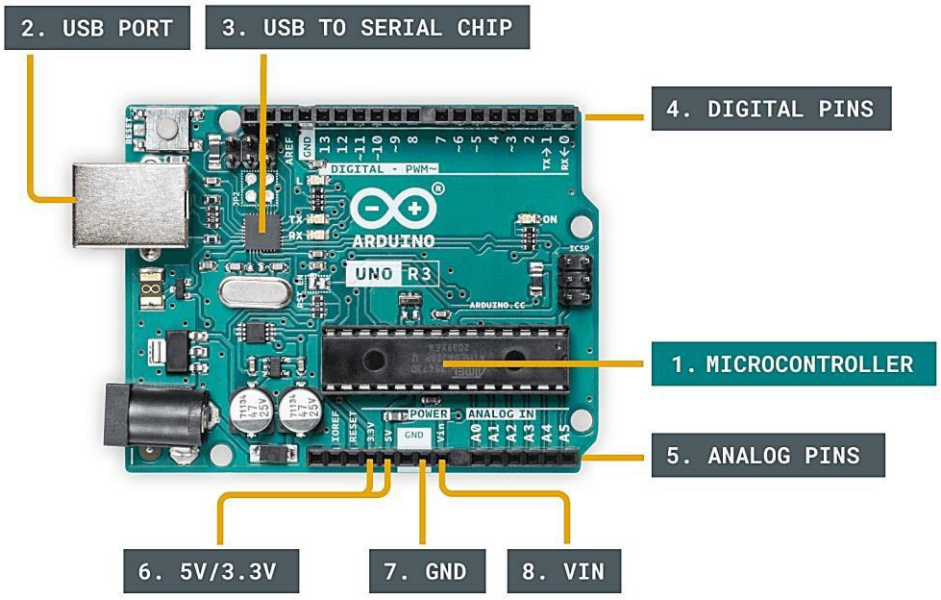
MKR Ailesi, herhangi bir ek devre olmadan muhteşem projeler oluşturmak için birleştirilebilen bir dizi kart, kalkan & taşıyıcıdır. Her kart, Wi-Fi®, Bluetooth®, LoRa®, Sigfox®, NB-IoT iletişimini etkinleştiren bir radyo modülüyle (MKR Zero hariç) donatılmıştır. Ailedeki tüm kartlar Arm® Cortex®-M0 32-bit SAMD21 düşük güç işlemcisine dayanmaktadır ve güvenli iletişim için bir kripto çipiyle donatılmıştır.
MKR Ailesi kalkanları & taşıyıcıları, kartın işlevlerini genişletmek için tasarlanmıştır: çevre sensörleri, GPS, Ethernet, motor kontrolü ve RGB matrisi gibi.

Mega Aile

Mega ailesinde, yüksek işlem gücü ve GPIO pinleri gerektiren projelere yönelik kartlar bulacaksınız.

Arduino Uno R3 Kartının Anatomisi

Tüm Arduino kartları birbirinden farklı olsa da hemen hemen her Arduino'da bulunabilen birkaç temel bileşen vardır. Aşağıdaki resme bir bakalım:



Arduino kartının temel bileşenleri.

- **1. Mikrodenetleyici** - Bu bir Arduino'nun beynidir ve içine programlar yüklediğimiz bileşendir. Bunu, yalnızca belirli sayıda şeyi yürütmek üzere tasarlanmış küçük bir bilgisayar olarak düşünün. Atmel Atmega 328P mikrodenetleyiciye sahiptir.
- **2. USB portu** - Arduino kartınızı bilgisayara bağlamak için kullanılır.

- **3. USB'den Seriye çip** - USB'den Seriye, örneğin bir bilgisayardan gelen verileri yerleşik mikrodenetleyiciye çevirmeye yardımcı olduğu için önemli bir bileşendir. Bu, Arduino kartını bilgisayarınızdan programlamanızı mümkün kılan şeydir.
- **4. Dijital pinler** - dijital mantık kullanan pinler (0,1 veya LOW/HIGH). Genellikle anahtarlar için ve bir LED'i açıp kapatmak için kullanılır.
- **5. Analog pinler** - 10 bit çözünürlükte (0-1023) analog değerleri okuyabilen pinler.
- **6. 5V / 3.3V pinleri** - bu pinler harici bileşenlere güç sağlamak için kullanılır.
- **7. GND** - ayrıca şu şekilde de bilinir ground, negative veya sadece - , elektrik seviyesi 0 volt olan bir devreyi tamamlamak için kullanılır.
- **8. VIN** - Harici güç kaynaklarını bağlayabileceğiniz Voltaj Girişi anlamına gelir.

Arduino kartına bağlı olarak çok daha fazla bileşen bulacaksınız. Yukarıda listelenen öğeler genellikle herhangi bir Arduino kartında bulunur.

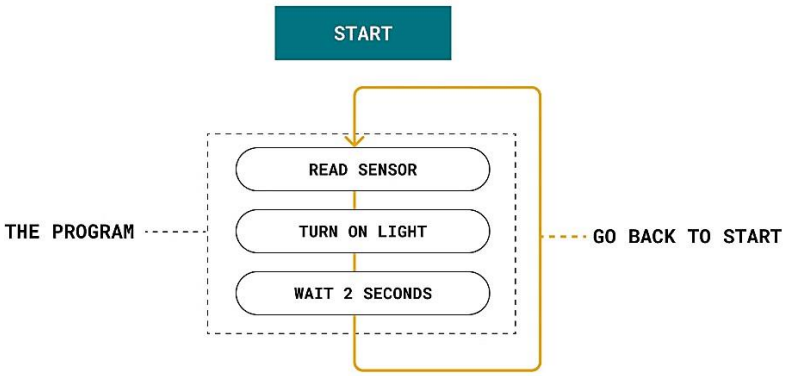
Temel İşlemler

Çoğu Arduino kartı, mikrodenetleyicide çalışan tek bir programa sahip olacak şekilde tasarlanmıştır. Bu program, bir LED'i yanıp söndürmek gibi tek bir eylemi gerçekleştirmek üzere tasarlanabilir. Ayrıca bir döngüde yüzlerce eylemi yürütmek üzere de tasarlanabilir. Kapsam bir programdan diğerine değişir.

Mikrodenetleyiciye yüklenen program, güç verildiği anda çalışmaya başlayacaktır. Her programın "loop" adlı bir fonksiyonu vardır. Döngü fonksiyonunun içinde örneğin şunları yapabilirsiniz:

- Bir sensörü oku.
- Bir ışık aç.
- Bir koşulun sağlanıp sağlanmadığını kontrol edin.
- Yukarıdakilerin hepsi.

Bir programın hızı, ona yavaşlamasını söylemediğimiz sürece inanılmaz derecede hızlıdır. Programın boyutuna ve mikrodenetleyicinin onu yürütmesinin ne kadar sürdüğüne bağlıdır, ancak genellikle **mikro saniyeler (saniyenin milyonda biri)** cinsindendir.



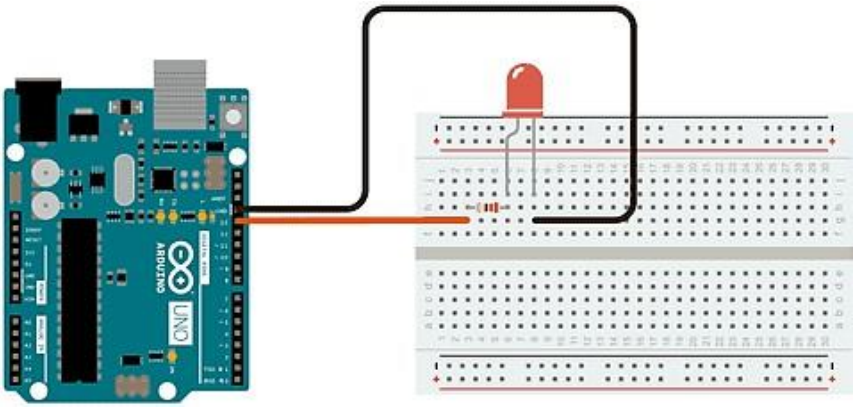
Arduino'nun temel işleyişi

Avrupa Birliği tarafından ortak finanse edilmektedir

Devre Temelleri

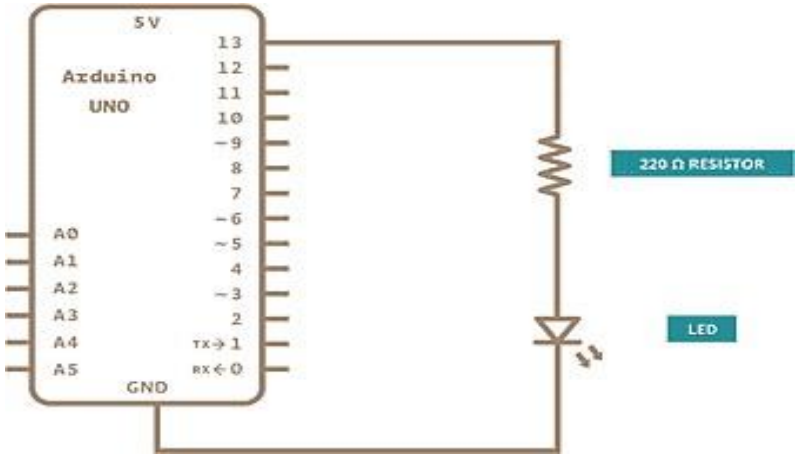
Devreler en az bir aktif elektronik bileşenden ve akımın geçebilmesi için teller gibi iletken bir malzemeden oluşur. Arduino ile çalışırken, çoğu durumda projeniz için bir devre inşa edeceksiniz.

Basit bir devre örneği, bir **LED devresidir**. Arduino'daki bir pinden bir tel, bir direnç (LED'i yüksek akımdan korumak için) aracılığıyla bir LED'e ve son olarak toprak pinine (GND) bağlanır. Pin YÜKSEK (HIGH) duruma ayarlandığında, Arduino kartındaki mikrodenetleyici devreden bir elektrik akımının geçmesine izin verir ve bu da LED'i açar. Pin DÜŞÜK (LOW) duruma ayarlandığında, devreden bir elektrik akımı geçmediği için LED kapanır.



Arduino ile LED devresi.

Devreler genellikle devrenizin planları olan şemalar olarak temsil edilir. Aşağıdaki görüntü, yukarıdaki görüntüde gösterilen aynı devrenin şematik temsilini gösterir.



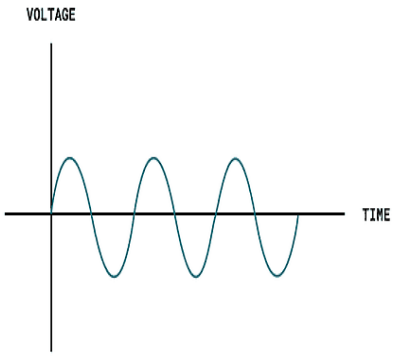
Bir devrenin şemaları.

Elektronik Sinyaller

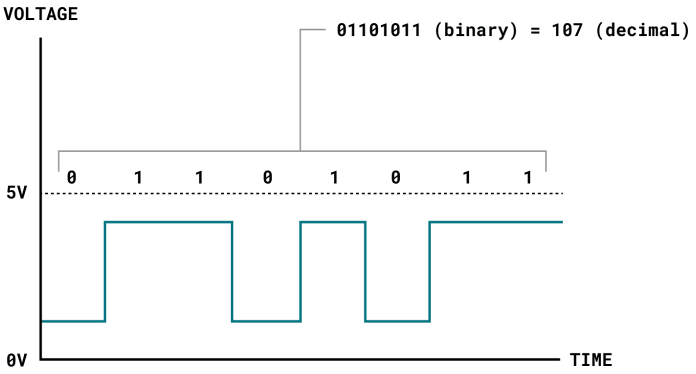
Elektronik bileşenler arasındaki tüm iletişim elektronik sinyallerle kolaylaştırılır**.** İki ana elektronik sinyal türü vardır: **analog ve dijital**.

Avrupa Birliği tarafından ortak finanse edilmektedir

Analog Sinyal



Dijital Sinyal



Analog sinyalin temelleri

Analog bir sinyal genellikle bir aralığa bağlıdır. Bir Arduino'da bu aralık tipik olarak 0-5V veya 0-3.3V'tur.

Örneğin bir potansiyometre (bir devrenin direncini değiştirmek için kullanılan analog bir bileşen) kullanırsak, bu aralığı (0-5V) elle ayarlayabiliriz. Programda bu, 10 bitlik bir çözünürlük olan 0-1023 aralığında gösterilir.

Analog sinyali Darbe Genişlik Modülasyonu (PWM) kullanarak yazarsak, 8 bit çözünürlük kullandığımız için 0-255 arasında bir aralık kullanabiliriz.

Dijital sinyalin temelleri

Dijital bir sinyal biraz farklı çalışır, programda yüksek (HIGH) veya düşük (LOW) durumlar olarak okunan yalnızca iki ikili durumu (0 veya 1) temsil eder. Bu, modern teknolojideki en yaygın sinyal türüdür.

Arduino üzerinde dijital sinyalleri kolayca okuyabilir ve yazabilirsiniz; bu, örneğin buton durumlarını okumak veya bir şeyi açıp kapatmak için kullanışlıdır.

Dijital sinyaller çok temel görünebilir (sadece 0 veya 1), ancak aslında çok daha gelişmiştir. Örneğin, yüksek veya düşük bir durumu birkaç kez hızla göndererek bir dizi oluşturabiliriz. Bu, **ikili dizi** veya **bit akışı** olarak bilinir.

İki ikili diziye bir göz atalım:

```
1- 101101
2- 101110001110011
```

Ondalık formatta şu şekildedir:

```
1- 45
2- 23667
```

Avrupa Birliği tarafından ortak finanse edilmektedir

Bu, yüksek ve düşük sinyalleri hızla göndererek büyük miktarda veriyi bir noktadan diğerine göndermenin akıllıca bir yoludur. Sinyallerden gelen verileri yorumlamak için **Seri İletişim Protokollerini** kullanırız.

Sensörler ve Aktüatörler

Arduino ile çalışırken sensörleri ve aktüatörleri ve aralarındaki farkları anlamak önemlidir.

Sensör Nedir?

Sensör, en basit tanımıyla, çevresini *algılamak*, yani fiziksel bir parametreyi, örneğin sıcaklığı kaydetmek ve bunu elektronik sinyale dönüştürmek için kullanılır.

Sensörler basit bir buton şeklinde de olabilir: Bir durum değiştiğinde (bir butona bastığımızda), elektronik sinyal düşükten yükseğe (0'dan 1'e) geçer.

Birçok sensör türü ve bunlardan veri kaydetmenin çeşitli yolları vardır. Belki de kullanımı en kolay olanı, bir Arduino analog pinine beslenen voltaj girişini (genellikle 0-5 volt arasında) değiştirerek bir dizi değeri ilettiğimiz analog sensördür. Bu size basitçe 0-1023 (10 bit çözünürlük) arasında bir aralık verir.

Bir kütüphaneyi kullandığımız birçok durumda ihtiyacımız olan tek şey bir satır koddur:

```
1 sensorValue = sensor.read();
```

Aktüatör Nedir?

Bir aktüatör, basit bir ifadeyle, fiziksel bir durumu harekete geçirmek veya değiştirmek için kullanılır. Bazı örnekler şunlardır:

- Bir ışık (örneğin bir LED).
- Motor.
- Bir anahtar.

Aktüatörler elektrik sinyallerini örneğin radyant enerjiye (ışık) veya mekanik enerjiye (hareket) dönüştürür.

Aktüatörlerin nasıl kontrol edildiği gerçekten sahip olduğumuz bileşen türüne bağlıdır. En basit yol bir şeyi basitçe açıp kapatmaktır, daha gelişmiş olanı ise bir bileşenin aldığı voltaj miktarını (yani bir motorun hızını) kontrol etmektir.

Aktüatörleri kontrol etmek için yaygın olarak kullanılır. digitalWrite() ve analogWrite().

```
1 digitalWrite(LED, HIGH); //turn on an LED
2 digitalWrite(LED, LOW);  //turn off an LED
3
4 analogWrite(motor, 255); //set a motor to maximum capacity
5 analogWrite(motor, 25);  //set a motor to 10% of its capacity
```

Giriş ve Çıkış

Sensörler ve aktüatörler, genellikle girişler ve çıkışlar olarak adlandırılır. Bir program yazdığımızda, bir sensörün durumunu kontrol eden ve bir şeyi harekete geçirip geçirmemesi gerektiğine karar veren koşullar oluşturmak yaygındır.

Bunun temel bir örneği bir **buton** ve bir **LED'dir.** Bir butona basılıp basılmadığını kontrol eden, LED'i açan ve butona basılmazsa kapatan bir koşul yazabiliriz. Bir Arduino programında, bu şu şekilde görünür:

```
1 int buttonState = digitalRead(buttonPin); //read and store the
                                             button state (0 or 1)
2
3 if(buttonState == HIGH){          //check if state is high (button
                                     is pressed)
4    digitalWrite(LED, HIGH);     //turn on LED
5 } else {
6    digitalWrite(LED, LOW);      //turn off LED
7 }
```

Seri İletişim Protokolleri

Yukarıda belirtilen dijital sinyalleri kullanarak veri gönderen birkaç seri iletişim protokolü vardır. En yaygın olanları **UART, SPI** ve **I²C'dir**. UART protokolü, diğer şeylerin yanı sıra, yeni bir program yüklemek veya doğrudan bir Arduino'dan veri okumak gibi bir bilgisayar ve Arduino kartı arasında veri göndermek için kullanılır.

SPI ve I²C protokolleri hem dahili hem de harici bileşenler arasındaki iletişim için kullanılır. İletişim, Arduino'daki belirli bir pine bağlı olan **seri veri yolu adı verilen bir şey tarafından gerçekleştirilir.**

I²C protokolünü kullanarak, aynı pin üzerinde birden fazla sensörü bağlayabilir ve verileri doğru bir şekilde alabiliriz. Her cihazın, veri istekleri yaparken kullandığımız programda belirtmemiz gereken bir adresi vardır.

Hafıza

"Standart" Arduino'nun genellikle iki belleği vardır: SRAM ve Flash bellek.

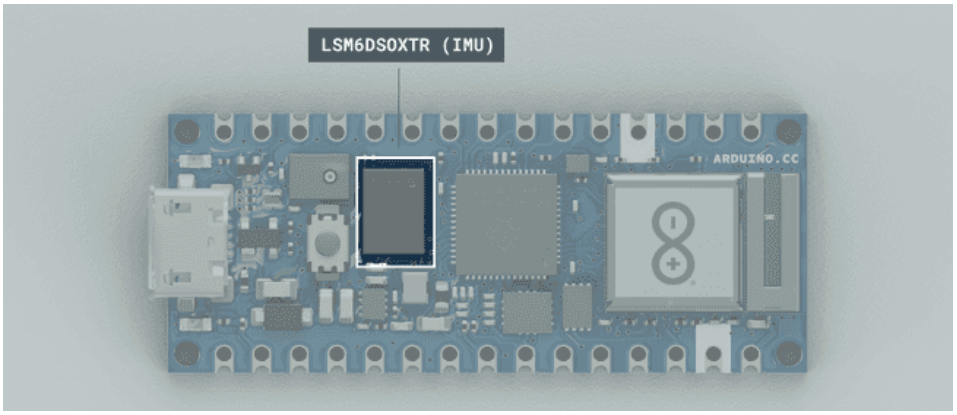
SRAM (Statik Rastgele Erişim Belleği), örneğin bir değişkenin değerini (örneğin bir Boolean'ın durumu) depolamak için kullanılır. Güç kapatıldığında, bu bellek sıfırlanır.

Flash bellek, öncelikle ana programı veya mikrodenetleyici talimatlarını depolamak için kullanılır. Bu bellek, güç kapatıldığında silinmez, böylece mikrodenetleyici talimatları, kart güç aldığı anda yürütülür.

Bir Arduino'da ne kadar bellek bulunduğu karttan karta değişir. Örneğin **Arduino UNO'da** 32kB flaş / 2kB SRAM bulunurken, **Nano 33 IoT'de 256kB flaş / 32kB SRAM bulunur.**

Avrupa Birliği tarafından ortak finanse edilmektedir

Gömülü Sensörler



Nano RP2040 Connect kartında bir IMU (Atalet Ölçüm Birimi).

Birçok yeni Arduino kartı gömülü sensörlerle donatılmıştır. Örneğin, **Nano 33 BLE Sense'in** 7 gömülü sensörü vardır ancak yalnızca **45x18 mm'dir** (bir baş parmağın boyutu). Bunların hepsi yukarıda belirtildiği gibi I²C protokolü aracılığıyla bağlanır ve benzersiz bir adrese sahiptir.

Arduino API

Arduino API, diğer adıyla "Arduino Programlama Dili", C/C++ diline dayalı çeşitli fonksiyon, değişken ve yapılardan oluşur.

Ana Parçalar

Arduino API'si üç ana bölüme ayrılabilir: **fonksiyonlar, değişkenler** ve **yapı** :

- **Fonksiyonlar:** Arduino kartını kontrol etmek ve hesaplamalar yapmak için. Örneğin, bir durumu dijital bir pine okumak veya yazmak, bir değeri eşlemek veya seri iletişim kullanmak için.

Değişkenler: Arduino sabitleri, veri tipleri ve dönüşümler. Örn. `int`, `boolean`, `array`.

- **Yapı:** Arduino (C++) kodunun öğeleri, örneğin:
  - taslak (loop(), setup())
  - kontrol yapısı (if, else, while, for)
  - aritmetik operatörler (çarpma, toplama, çıkarma)
  - karşılaştırma operatörleri , örneğin == (eşittir), != (eşit değil), > (büyüktür).

Arduino API, Arduino donanımını kontrol etmek için birçok ekleme içeren, C++ programlama dilinin basitleştirilmiş hali olarak tanımlanabilir.

Program Yapısı

Bir Arduino programının mutlak minimum gereksinimi iki fonksiyonun kullanılmasıdır:

`void setup()` ve `void loop()` "void" ifadesi yürütme sırasında hiçbir şeyin döndürülmediğini belirtir.

Avrupa Birliği tarafından ortak finanse edilmektedir

- `void setup()`- Bu fonksiyon yalnızca Arduino açıkken bir kez yürütülür. Burada bir pinin modu (giriş veya çıkış), seri iletişimin baud hızı veya bir kütüphanenin başlatılması gibi şeyleri tanımlarız.
- `void loop()`- Burada, örneğin bir lambanın bir girdiye bağlı olarak açılıp/kapatılması veya her X saniyede bir sensör okuması yapılması gibi, tekrar tekrar çalıştırmak istediğimiz kodu yazıyoruz.

Yukarıdaki fonksiyonlar bir Arduino taslağında her zaman gereklidir, ancak elbette daha uzun programlar için kullanışlı olan birkaç fonksiyon daha ekleyebilirsiniz.

"Taslak" (Sketch)

Arduino projesinde bir programa taslak "sketch" denir. Taslak, programınızı içine yazdığınız bir dosyadır. `.ino` uzantısına sahiptir ve her zaman aynı isimli klasörde saklanır.

Klasör, çiziminize dahil edilebilecek **başlık dosyası gibi diğer dosyaları da içerebilir.**

Örnek Taslak

Aşağıda bazı popüler Arduino programlama öğelerini içeren standart bir Arduino taslağının örneği yer almaktadır.

```
/*
This is a comment at the top of a program,
it will not be recognized as code. Very good
to add an explanation of what your code does
here.

This sketch shows how to read a value from a
sensor connected to pin A1, print it out in
the Serial Monitor, and turn on an LED connected
to pin number 2 if a conditional is met.
*/

int sensorPin = A1; //define pin A1 (analog pin)
int ledPin = 2; //define pin 2 (digital pin)
int sensorValue; //create variable for storing readings

//void setup is for configurations on start up
void setup() {
    Serial.begin(9600); //initialize serial communication
    pinMode(ledPin, OUTPUT); //define ledPin as an output
}

void loop() {
    sensorValue = analogRead(sensorPin); // do a sensor reading

    Serial.print("Sensor value is: "); //print a message to
                                       the serial monitor
```

```
27      Serial.println(sensorValue); //print the value to the
                                        serial monitor
28
29      //check if sensorValue is below 200
30      if(sensorValue < 200) {
31          digitalWrite(ledPin, HIGH); //if it is, turn on the
                                          LED on pin 2.
32      }
33      //if sensorValue is above 200, turn off the LED
34      else{
35          digitalWrite(ledPin, LOW);
36      }
37  }
```

Kütüphaneler

Arduino kütüphaneleri, standart Arduino API'nin bir uzantısıdır ve hem resmi hem de topluluk tarafından katkıda bulunulan **binlerce kütüphaneden oluşur.**

Kütüphaneler, belirli bir sensörü okumak, bir motoru kontrol etmek veya İnternete bağlanmak gibi karmaşık kodların kullanımını basitleştirir. Tüm bu kodları kendiniz yazmak yerine, bir kütüphane yükleyebilir, onu kodunuzun en üstüne ekleyebilir ve kütüphanenin mevcut işlevlerinden herhangi birini kullanabilirsiniz. Tüm Arduino kütüphaneleri açık kaynaklıdır ve **herkes tarafından ücretsiz olarak kullanılabilir.**

Bir kütüphaneyi kullanmak için onu kodunuzun en üstüne eklemeniz gerekir, aşağıdaki örnekte olduğu gibi:

```
1 #include <Library.h>
```

Arduino Yazılım Araçları

Arduino ekosisteminin bir diğer ayrılmaz parçası da yazılım araçlarıdır.

Yaygın olarak bilinen adıyla Arduino IDE, **entegre bir geliştirme ortamıdır.** Peki bu tam olarak ne anlama geliyor?

Kartınızı programlamak için bir program yazmanız, bu programı makine koduna derlemeniz ve son olarak yeni programı kartınıza göndermeniz gerekir.

Arduino IDE, yazılan ilk kod satırından Arduino kartının mikrodenetleyicisinde çalıştırılmasına kadar tüm bunları kolaylaştırır. Tüm kod geliştirmelerinizi yönetmek için indirebileceğiniz (veya çevrimiçi bir sürümünü kullanabileceğiniz) bir program veya uygulamadır. Eskiden, bu karmaşık bir süreçti ve elektronik ve bilgisayar biliminde iyi bir bilgi seti gerektiriyordu. Şimdi, Arduino IDE'nin yardımıyla **herkes bunu nasıl yapacağını öğrenebilir.**

Avrupa Birliği tarafından ortak finanse edilmektedir

Günümüzde üç adet Arduino IDE'si mevcuttur:

- Arduino IDE 1.8.x (klasik)
- Arduino IDE 2 (yeni)
- Arduino Bulut Editörü (çevrimiçi)

Tipik Bir İş Akışı

IDE kullanarak Arduino kartına kod yüklemek için genellikle aşağıdakiler yapılır:

1. Kartınızı kurun - bu, kartınız için doğru "paketi" kurmak anlamına gelir. Paket olmadan kartınızı kullanamazsınız. Kurulum doğrudan IDE'de yapılır ve hızlı ve kolay bir işlemdir.

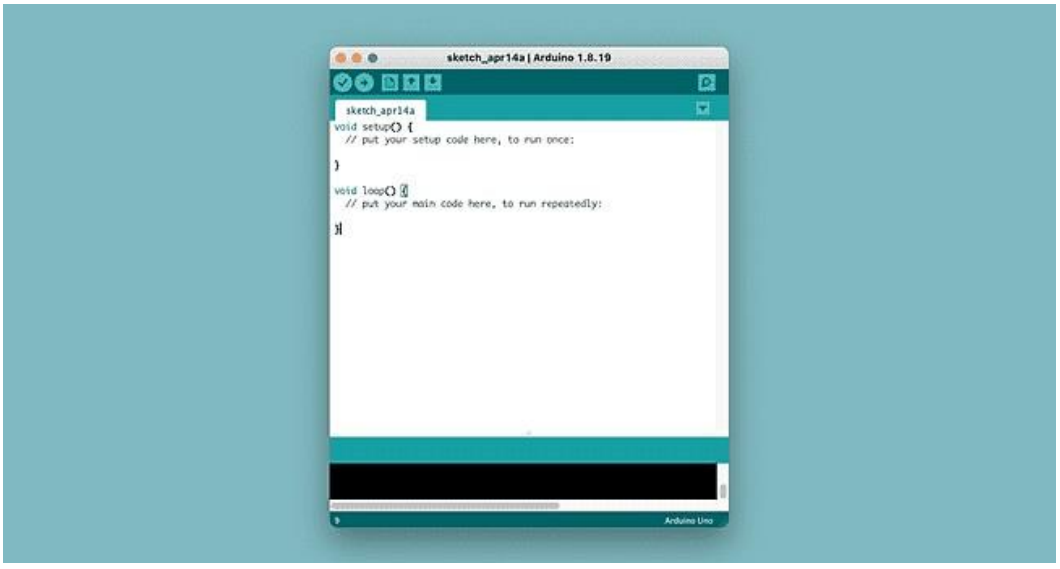
2. Yeni bir taslak oluşturun - taslak, ana program dosyanızdır. Burada mikrodenetleyicide yürütmek istediğimiz bir dizi talimat yazıyoruz.

3. Çiziminizi derleyin - yazdığımız kod Arduino'muza yüklendiğinde tam olarak göründüğü gibi değildir: kodu derlemek, hataları kontrol ettiğimiz ve onu ikili bir dosyaya (1'ler ve 0'lar) dönüştürdüğümüz anlamına gelir. Bir şey başarısız olursa, hata konsolunda bunu alırsınız.

4. Çiziminizi yükleyin - derleme başarılı olduğunda, kod panonuza yüklenebilir. Bu adımda, panoyu fiziksel olarak bilgisayara bağlarız ve doğru seri portu seçeriz.

5. Seri Monitör (isteğe bağlı) - çoğu Arduino projesi için, kartınızda neler olup bittiğini bilmek önemlidir. Tüm IDE'lerde bulunan Seri Monitör aracı, kartınızdan bilgisayarınıza veri gönderilmesine olanak tanır.

Arduino IDE 1.8.x



Klasik Arduino IDE'si

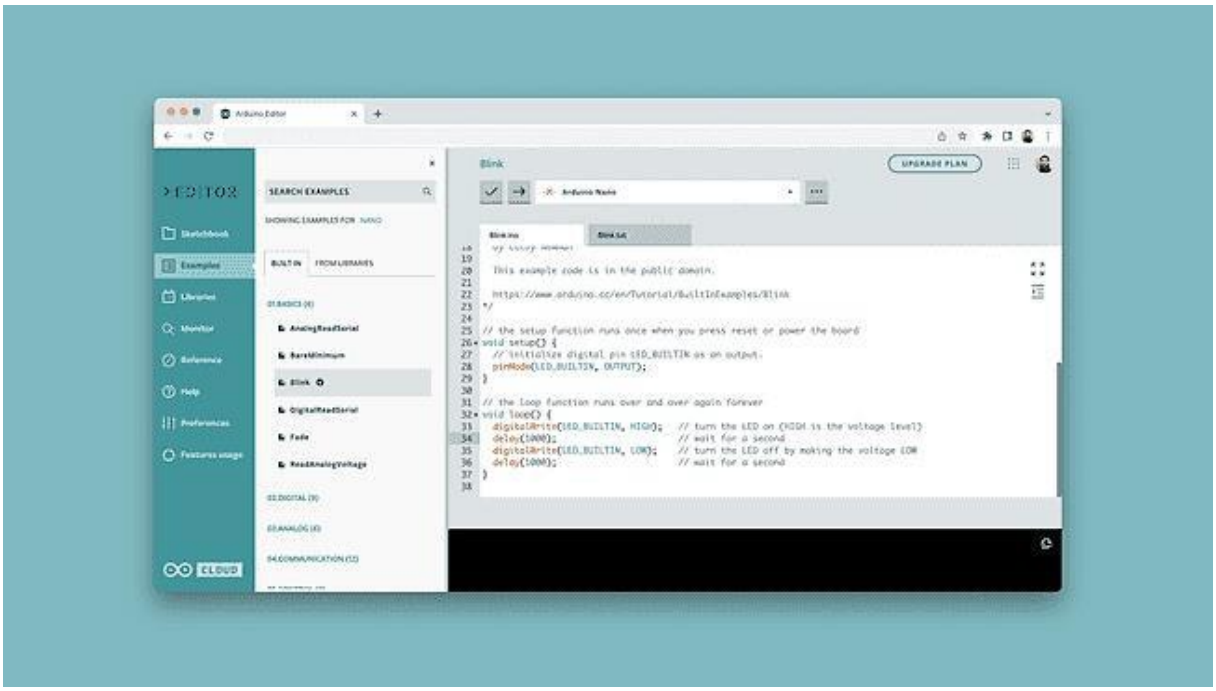
Günümüzde "eski" editör olarak kabul edilen Arduino IDE 1.8.X veya "Java IDE", Arduino'nun ilk piyasaya sürüldüğü zamanlarda yayınlanan editördür.

Avrupa Birliği tarafından ortak finanse edilmektedir

Arduino IDE2

Yeni Arduino IDE'si

Bulut Editörü

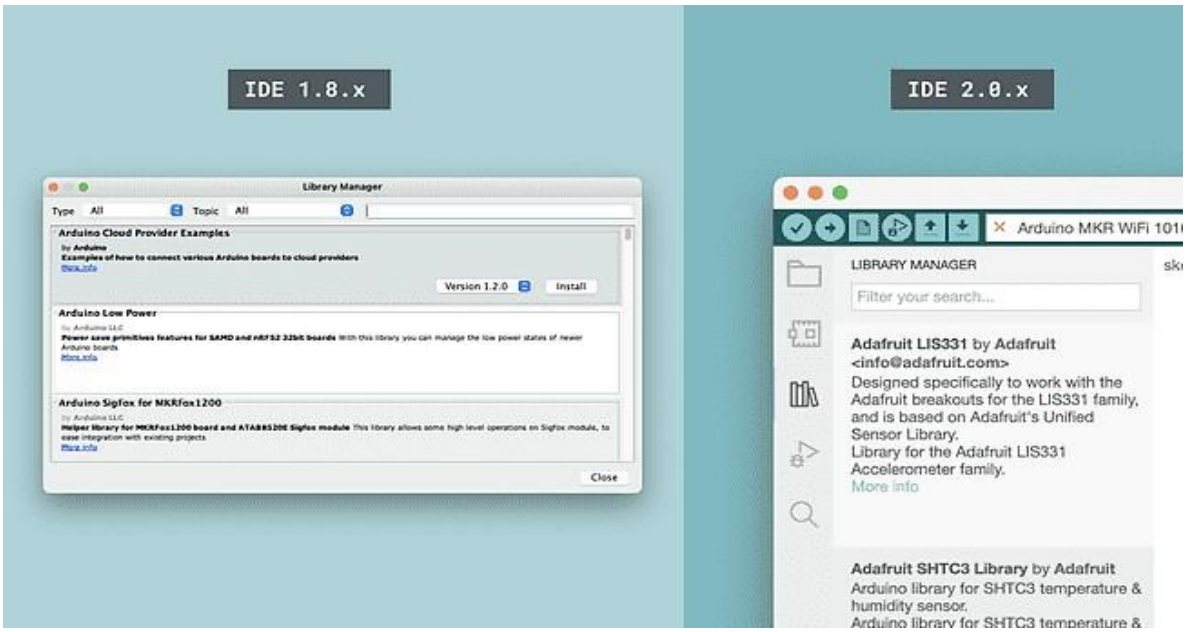


Bulut Editörü

Arduino **Cloud Editor** , Arduino Cloud paketinin bir parçası olan çevrimiçi bir IDE'dir. İşlevsel olarak benzer olan bu düzenleyici, diğer özelliklerin yanı sıra çevrimiçi depolama ile tamamen web tabanlıdır. Cloud Editor' u kullanmak için bir Arduino hesabı kaydetmeniz gerekecektir.

Avrupa Birliği tarafından ortak finanse edilmektedir

Kütüphane Yöneticisi



IDE 1.8.x ve IDE 2'deki kütüphane yöneticisi

IDE'nin her sürümünde Arduino yazılım kütüphanelerini yüklemek için bir kütüphane yöneticisi bulunur. Hem resmi hem de katkıda bulunulan kütüphaneler olmak üzere binlerce kütüphane doğrudan indirilebilir. Her kütüphane için kod örnekleri indirme sırasında kullanılabilir hale getirilir.

Arduino Yazılımını (IDE) Kullanma

Çevrimdışı IDE, internet bağlantısı olmadan kod yazmayı ve onu panoya yüklemeyi kolaylaştırır.

Arduino Yazılımı (IDE), kod yazmayı ve onu çevrimdışı olarak panoya yüklemeyi kolaylaştırır. İnternet bağlantısı zayıf veya hiç olmayan kullanıcılar için bunu öneriyoruz. Bu yazılım herhangi bir Arduino kartıyla kullanılabilir.

Şu anda Arduino IDE'nin iki sürümü var, biri IDE 1.xx ve diğeri IDE 2.x. IDE 2.x, IDE 1.xx'ten daha hızlı ve daha güçlü olan yeni bir ana sürümdür. Daha modern bir düzenleyici ve daha duyarlı bir arayüze ek olarak, kullanıcıların kodlama ve hata ayıklama konusunda yardımcı olmak için gelişmiş özellikler içerir.

Aşağıdaki adımlar çevrimdışı IDE'yi kullanırken size yol gösterebilir (IDE 1.xx veya IDE 2.x'i seçebilirsiniz):

1. Arduino Yazılım IDE'sini indirin ve kurun:

- **Arduino IDE 1.xx** (Windows , Mac OS , Linux , Windows ve Linux için Taşınabilir IDE , ChromeOS).
- Arduino IDE 2.x

2. Arduino kartınızı cihazınıza bağlayın.

3. Arduino Yazılımını (IDE) açın.

Arduino Entegre Geliştirme Ortamı - veya Arduino Yazılımı (IDE) - programları yüklemek ve onlarla iletişim kurmak için Arduino kartlarına bağlanır. Arduino Yazılımı (IDE) kullanılarak yazılan programlara **taslaklar** denir. Bu taslaklar metin düzenleyicide yazılır ve. .ino dosya uzantısıyla kaydedilir.

Çevrimdışı IDE 1.xx'i kullanma

Editör dört ana alandan oluşur:

1. Ortak işlevler ve bir dizi menü için düğmeler içeren bir Araç Çubuğu. Araç çubuğu düğmeleri programları doğrulamanıza ve yüklemenize, taslaklar oluşturmanıza, açmanıza ve kaydetmenize ve seri monitörü açmanıza olanak tanır.

2. Mesaj **alanı**, kaydetme ve dışa aktarma sırasında geri bildirim verir ve ayrıca hataları gösterir.

3. Kodunuzu yazmak için kullanacağınız **metin editörü**.

4. Metin **konsolu,** Arduino Yazılımı (IDE) tarafından üretilen metin çıktısını, tam hata mesajlarını ve diğer bilgileri de içerecek şekilde görüntüler.

Pencerenin sağ alt köşesinde yapılandırılmış kart ve seri port görüntülenir.

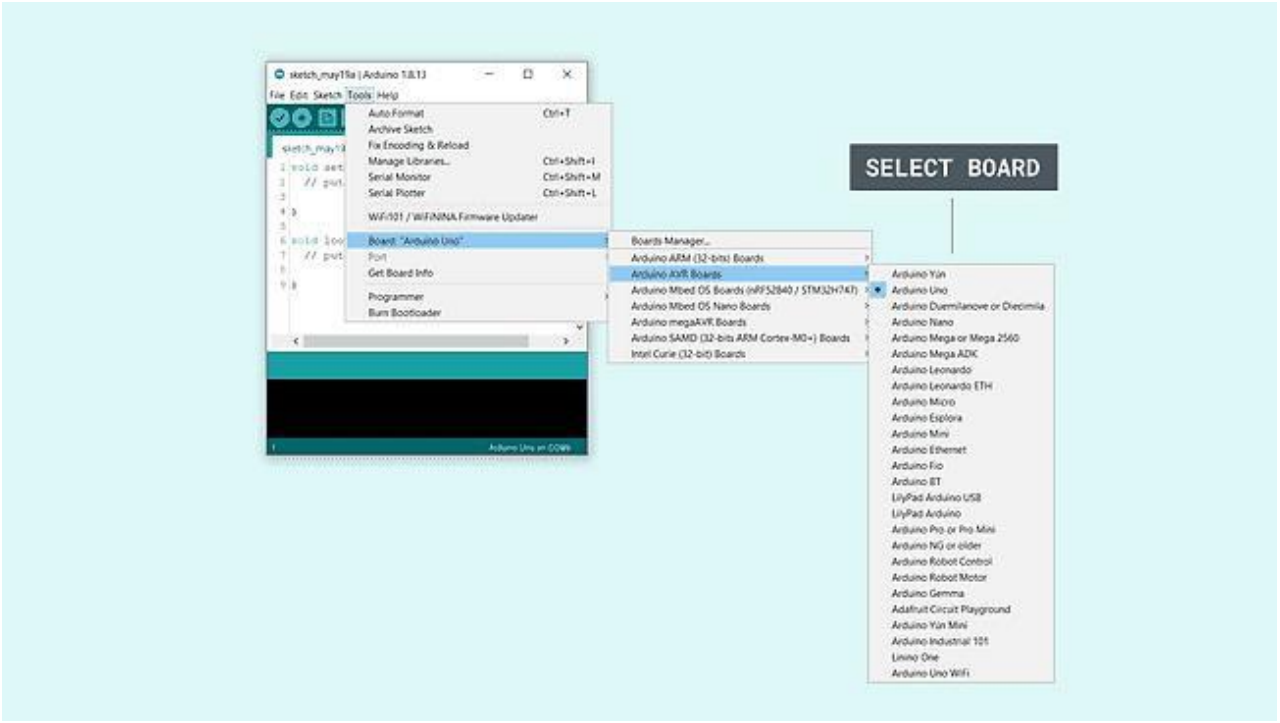
Arduino Yazılım IDE'si

Artık her şey hazır olduğuna göre, tahtanızın göz kırpmasını sağlamayı deneyelim!

5. Arduino veya Genuino kartınızı bilgisayarınıza **bağlayın**.

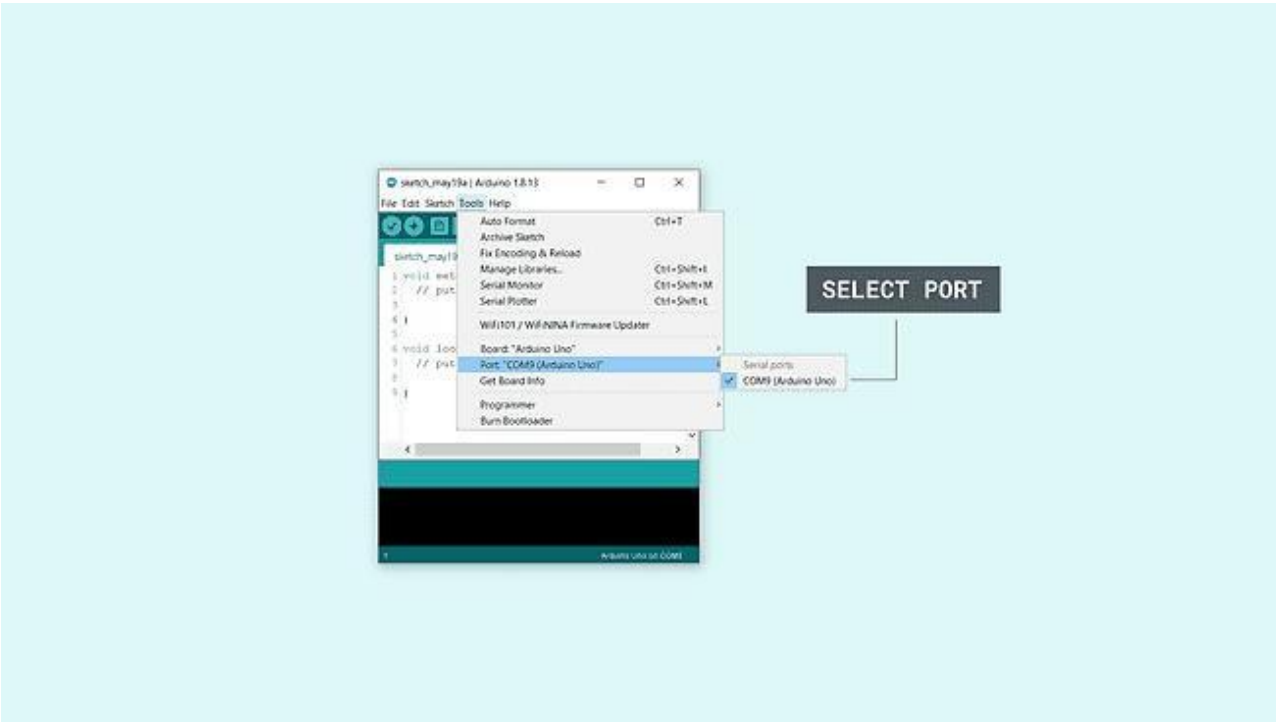
6. Şimdi doğru çekirdeği ve kartı seçmeniz gerekiyor. Bu, **Araçlar> Kart> Arduino AVR Kartları> Kart'a** giderek yapılır. Kullandığınız kartı seçtiğinizden emin olun. Kartınızı bulamazsanız, **Araçlar> Kart> Kart Yöneticisinden** ekleyebilirsiniz.

Avrupa Birliği tarafından ortak finanse edilmektedir



Bir kurul seçimi

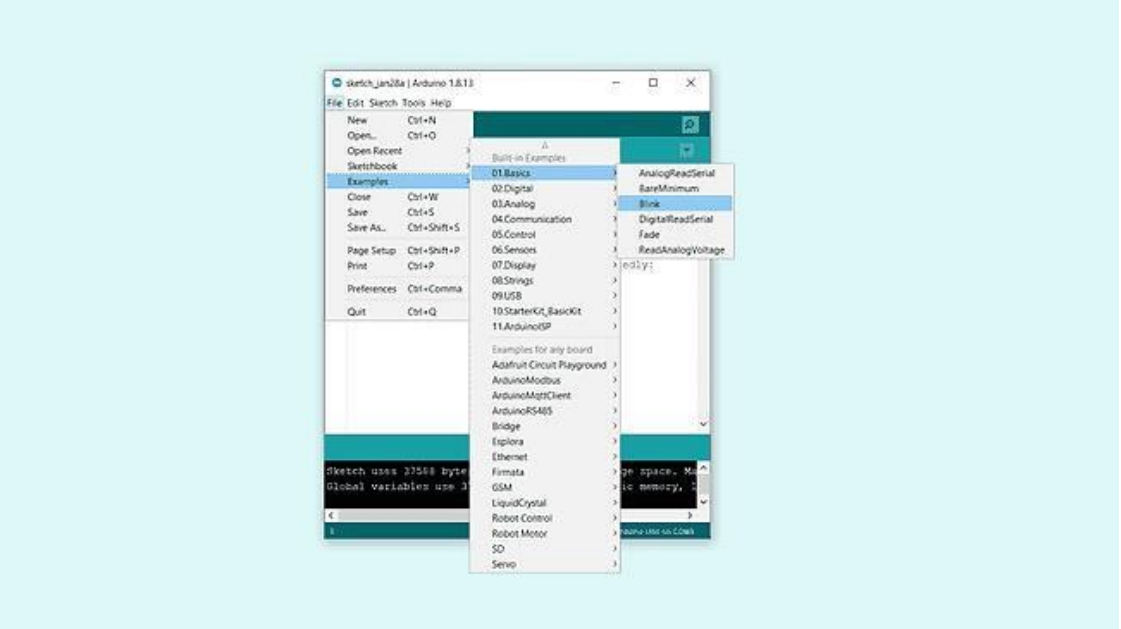
7. Şimdi, portu seçerek kartınızın bilgisayar tarafından bulunduğundan emin olalım. Bu, basitçe **Araçlar> Port'a** giderek yapılır, burada listeden kartınızı seçersiniz.



Port seçimi

Avrupa Birliği tarafından ortak finanse edilmektedir

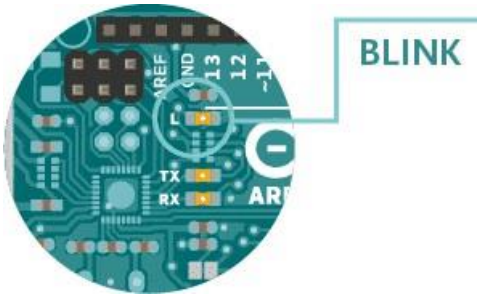
8. Bir örnek deneyelim: **Dosya> Örnekler> 01.Temeller> Göz kırpma' ya** gidin.



Bir örnek açmak

9. Bunu panonuza yüklemek için, sol üst köşedeki oka tıklamanız yeterlidir. Bu işlem birkaç saniye sürer ve bu işlem sırasında panonun bağlantısını kesmemek önemlidir. Yükleme başarılı olursa, alt çıktı alanında "Yükleme tamamlandı" mesajı görünecektir.

10. Yükleme tamamlandıktan sonra, kartınızda yanında L bulunan sarı LED'in yanıp sönmeye başladığını görmelisiniz. Parantez içindeki gecikme sayısını 100 olarak değiştirerek **yanıp sönme hızını ayarlayabilir ve Blink taslağını tekrar yükleyebilirsiniz. Şimdi LED çok daha hızlı yanıp sönmelidir.**



Tebrikler! Kartınızı yerleşik LED'ini yakıp söndürmek üzere başarıyla programladınız!

Çevrimdışı IDE 2.x'i kullanma

Editör dört ana alandan oluşur:

1. Ortak işlevler ve bir dizi menü için düğmeler içeren bir araç çubuğu. Araç çubuğu düğmeleri programları doğrulamanıza ve yüklemenize, taslaklar oluşturmanıza, açmanıza ve kaydetmenize, kartınızı ve portunuzu seçmenize ve seri monitörü açmanıza olanak tanır.

Avrupa Birliği tarafından ortak finanse edilmektedir

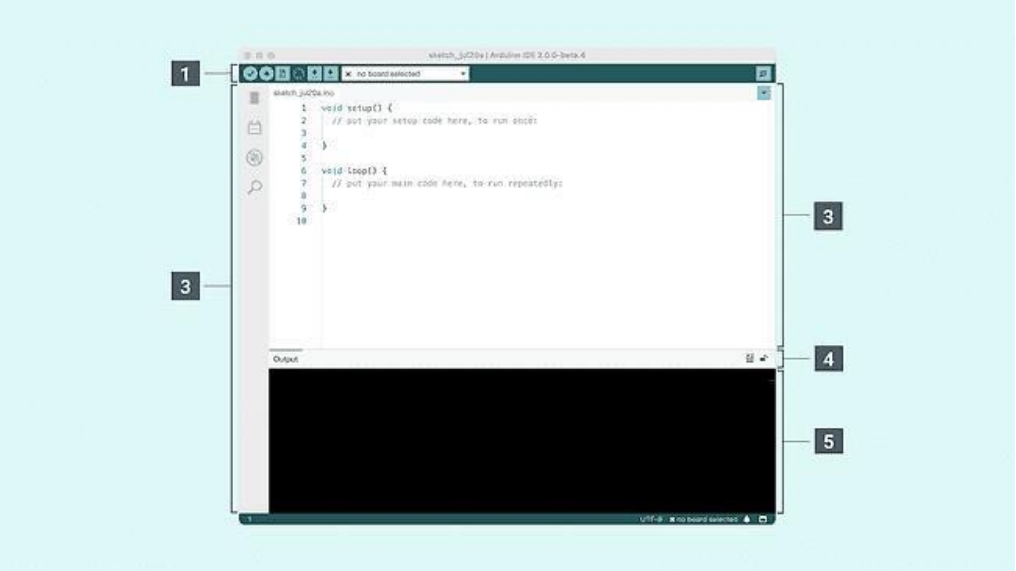
2. Düzenli olarak kullanılan araçlar için Kenar Çubuğu. Pano yöneticilerine, kütüphanelere, panonuzu hata ayıklamaya ve ayrıca bir arama ve değiştirme aracına hızlı erişim sağlar.

3. Kodunuzu yazmak için kullanacağınız **metin editörü**.

4. Konsol kontrolleri konsoldaki çıktı üzerinde kontrol sağlar.

5. Metin **konsolu,** Arduino Yazılımı (IDE) tarafından üretilen metin çıktısını, tam hata mesajlarını ve diğer bilgileri de içerecek şekilde görüntüler.

Pencerenin sağ alt köşesinde yapılandırılmış kart ve seri port görüntülenir.

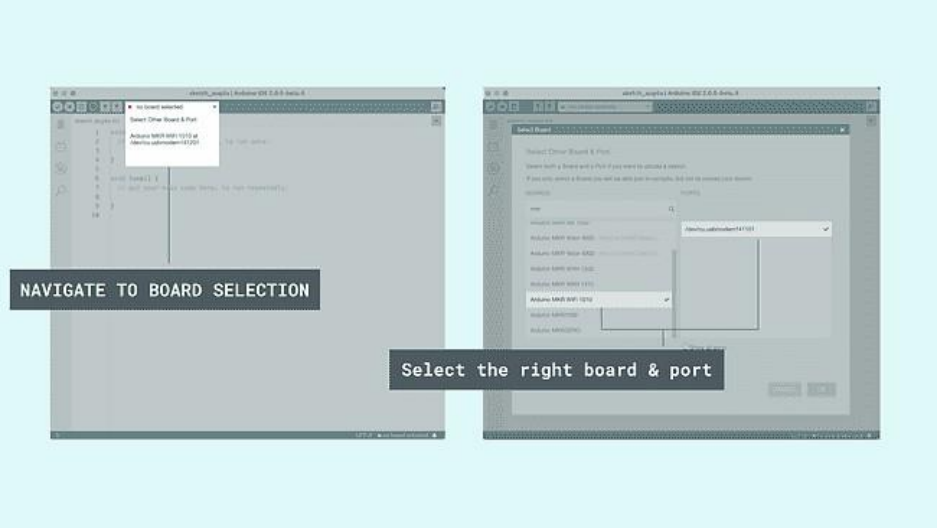


Arduino Yazılım IDE'si

Artık her şey hazır olduğuna göre, **tahtanızın göz kırpmasını sağlamayı deneyelim!**

1. Arduino veya Genuino kartınızı bilgisayarınıza **bağlayın.**

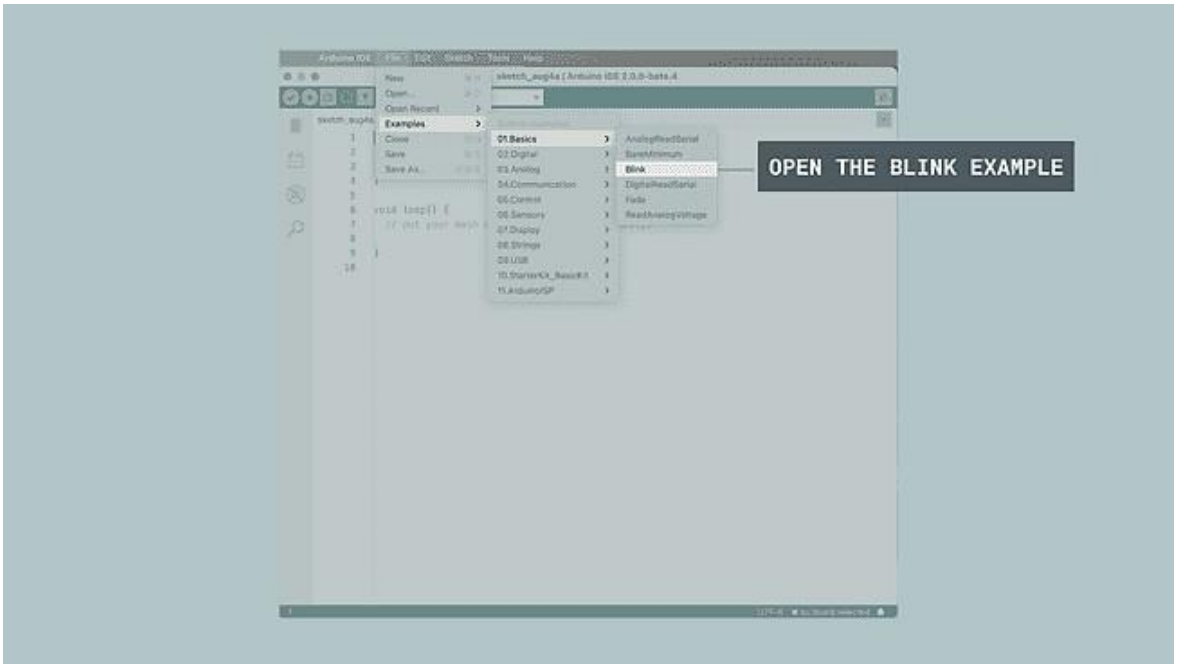
2. Şimdi doğru kartı ve portu seçmeniz gerekiyor. Bu araç çubuğundan yapılır. Kullandığınız kartı seçtiğinizden emin olun. Kartınızı bulamazsanız, kenar çubuğundaki kart yöneticisinden ekleyebilirsiniz.



Bir tahta ve port seçimi

Avrupa Birliği tarafından ortak finanse edilmektedir

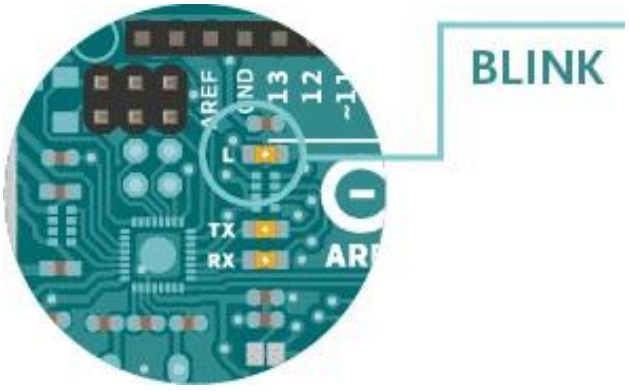
3. Bir örnek deneyelim: **Dosya> Örnekler> 01.Temeller> Göz kırpma' ya** gidin.



Bir örnek açmak

4. Bunu panonuza yüklemek için, sol üst köşedeki oka tıklamanız yeterlidir. Bu işlem birkaç saniye sürer ve bu işlem sırasında panonun bağlantısını kesmemek önemlidir. Yükleme başarılı olursa, alt çıktı alanında "Yükleme tamamlandı" mesajı görünecektir.

5. Yükleme tamamlandığında, kartınızda yanında **L** harfi bulunan sarı LED'in yanıp sönmeye başladığını görmelisiniz. Parantez içindeki gecikme sayısını 100 olarak değiştirerek **yanıp sönme hızını ayarlayabilir ve Blink taslağını tekrar yükleyebilirsiniz. Şimdi LED çok daha hızlı yanıp sönmelidir.**



Tebrikler! Kartınızı yerleşik LED'ini yakıp söndürmek üzere başarıyla programladınız!

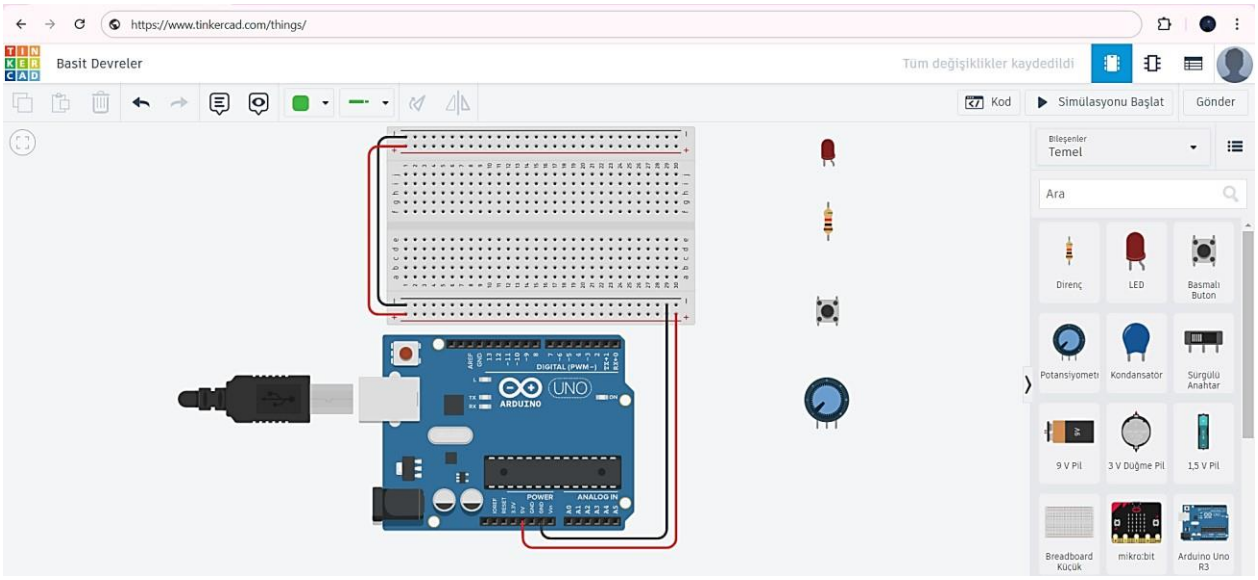
Bunların dışında kodlamasını kolayca yapabileceğimiz ve kolayca arduino veya elektronik devreler simüle edebileceğimiz **tinkercad** ismiyle ücretsiz bir çevrimiçi internet sitesi bulunmaktadır.

Avrupa Birliği tarafından ortak finanse edilmektedir

Tinkercad

Tinkercad, 3 boyutlu tasarım (3D), elektronik devre ve kodlama yapabileceğimiz eğitim amaçlı çevrimiçi ücretsiz bir Autodesk uygulamasıdır. Tinkercad kullanmak için bir program indirmeye gerek yoktur. Bulut tabanlı olarak çalışır böylece yaptığımız uygulamaları internet ağ bağlantısı olan herhangi bir PC vb. yardımı ile dilediğimiz yerden açıp kullanabiliriz.

Tinkercad Circuits bölümü, basit elektronik devre kurulumunu anlatmak için kullanılabilecek sade, görsellik anlamında etkileyici bir online simülasyon programıdır. Bu bölümde Tinkercad Circuits üzerinden temel elektronik ve Arduino uygulamaları, simülasyon kullanarak yapılabilir. Bu sayede herhangi bir devre elemanı ve kart kullanmadan robotik uygulamaların nasıl çalıştığını görebiliriz. Ayrıca oluşabilecek hataların önüne geçerek maddi kayıpları da engellemiş oluruz.



Tinkercad Circuits (Devreler)

Tinkercad'e Üye Olma ve Giriş

Tinkercad sitesi üzerinden tasarım oluşturmak veya Circuits ile simülasyon oluşturabilmek için sisteme kayıt olunması gerekmektedir. https://www.tinkercad.com adresine girdikten sonra Hemen Katıl seçeneğini seçerek sisteme üye olunabilir.

Hemen Katıl seçeneğini seçtikten sonra şekildeki gibi 3 seçenek ekrana gelir.

Avrupa Birliği tarafından ortak finanse edilmektedir

Eğitimciler, buradan başlayın: Bu seçenek ile sisteme kayıt olmak istendiğinde Öğretmen Sözleşmesi kabul edilerek öğretmen hesabı oluşturulabilir.

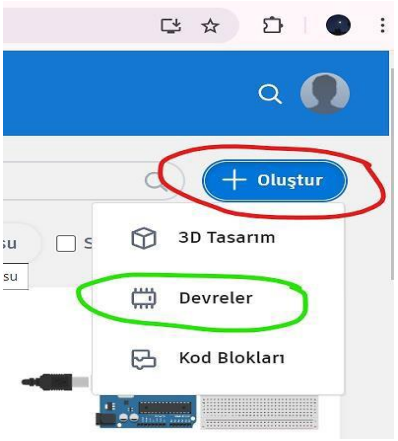
Öğretmenin oluşturduğu hesap üzerinden sınıflar tanımlanarak öğrencilerin sisteme kayıt olmadan, kod kullanarak giriş yapması sağlanabilir.

Öğrenciler, bir Sınıfa katılın: Bu seçenek sayesinde; öğretmen hesabı ile öğrenciler için oluşturulan ders kodları kullanılarak, sisteme kayıt olmadan giriş yapmaları sağlanabilir.

Kişisel Hesap Oluştur: Bu seçenek ile sistem üzerinde standart bir hesap oluşturarak sistem kullanılabilir. Eğer bir sınıf hesabı oluşturulmayacaksa bu seçenek seçilebilir.

Farklı bir cihazdan veya farklı bir zamanda sisteme giriş yapmak istenildiğinde var olan hesap ile sisteme girmek için ana sayfada bulunan Giriş yap seçeneği ile sisteme giriş yapılır.

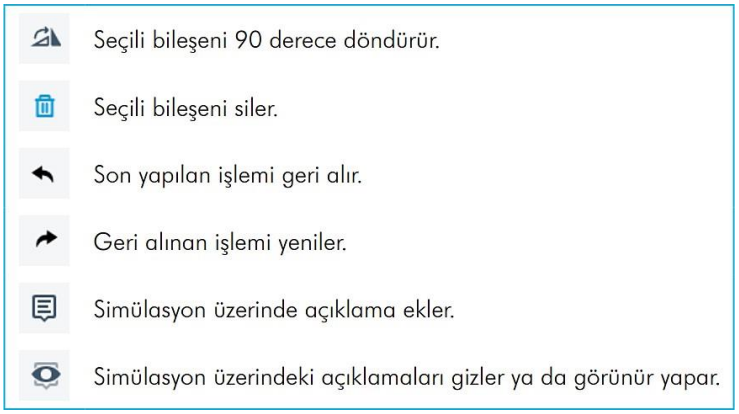
Sisteme giriş yapıldıktan sonra Tinkercad uygulaması kullanılır. Circuits (devreler) özelliğini kullanarak elektronik simülasyonlarını yapabilmek için açılan sayfada sol tarafta bulunan +oluştur düğmesine tıklayarak Devreler seçeneği seçilir.



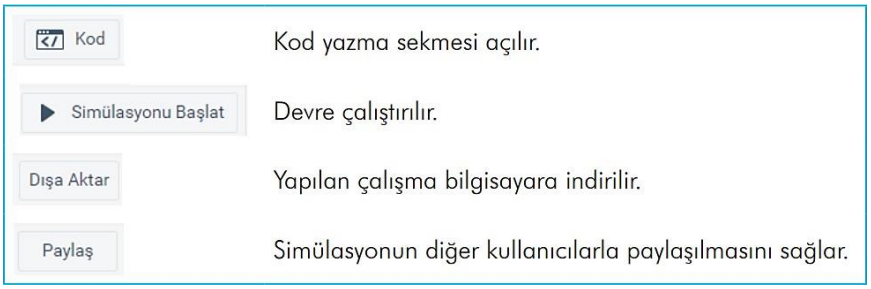
Avrupa Birliği tarafından ortak finanse edilmektedir

Yeni Devre Oluşturma

Devreler seçeneği seçildikten sonra yeni devreler oluşturmak için simülasyon sayfasına ulaşılır. Simülasyon ekranı üzerindeki bölümler aşağıdaki gibidir.



Tinkercad kısayolları

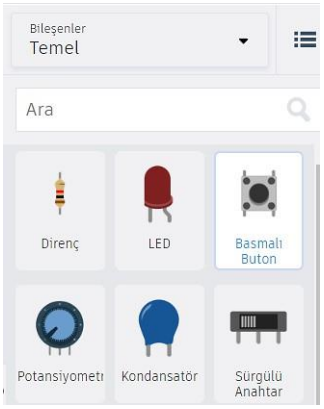


Simülasyon araçları

Devre sayfasını isimlendirmek için sayfanın sol üstünde bulunan Tinkercad simgesinin yanında bulunan yazıya tıklanarak proje ismi değiştirilebilir.

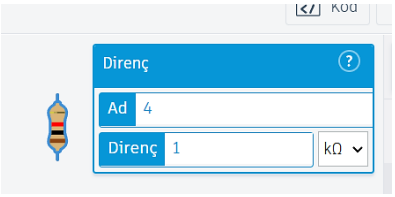
Bileşen Ekleme

Tasarım ekranına bileşen eklemek için sayfanın sağında bulunan menüden istenilen bileşen sürüklenerek ekranın ortasına konumlandırılabilir. Sayfa açıldığında temel bileşenler görüntülenmektedir.



Avrupa Birliği tarafından ortak finanse edilmektedir

Bileşen sahneye sürüklediğinde artık kullanılabilir hâle gelir ayrıca sağ üst taraftaki kutucuktan bileşene isim verilebilir.

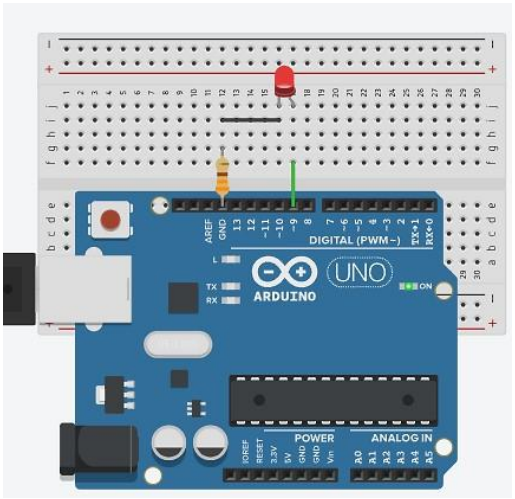


Bileşenler Arasında Bağlantı Kurma

Gerçek elektronik devreler tasarlanırken bileşenler arasında bağlantı kurmak için günlük hayatta lehim yapılabilir ya da lehime ihtiyaç duyulmadan breadboard kullanılarak jumper kablolar vasıtası ile bileşenler birbirlerine bağlanabilir. Simülasyon üzerinde bağlantı kurmak içinde breadboard ve bağlantı kabloları kullanılmalıdır. Bileşenler arasında bağlantı kurmak için bileşenin bacağı üzerine farenin sol tuşunu kullanarak kablo oluşturabilir, oluşturulan kablo fare hareket ettirilerek istenilen yere konumlandırılabilir.

Devre Bağlantısı Oluşturma

Şekildeki gibi bir devre oluştururken, Led'in kısa bacağı 330 ohm değerindeki direnç üzerinden Arduino'nun GND pinine bağlanmalı, uzun bacak ise Arduino'nun dijital pinlerinden herhangi birisine bağlanabilir. Bu durumda uzun bacak Arduino'nun hangi pinine bağlandıysa kodlama işleminin ona göre yapılması gerekmektedir.

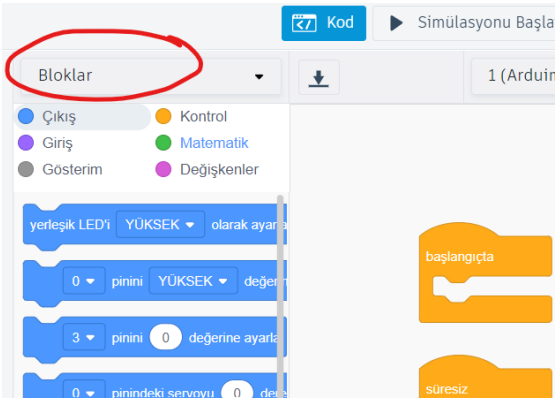


Kod Bloklarının Oluşturulması

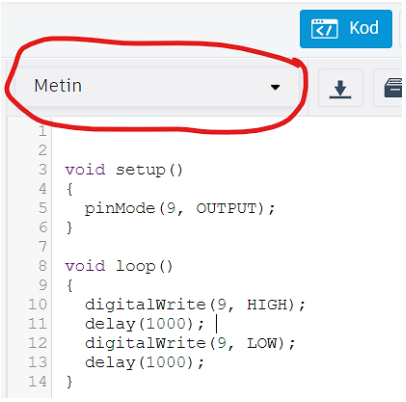
Devre bağlantısı kurulduktan sonra kodların oluşturulması gerekmektedir. Arduino kartı yazılan kodlara göre çalışacaktır. Bu nedenle kodların, yapılan bağlantıya uygun olarak oluşturulması gerekir. Kod bloklarını oluşturabilmek için sayfa üzerinde bulunan Kod sekmesine tıklanması yeterlidir. Bu işlemden sonra kod blokları paneli açılacaktır.

Ayrıca kodlama bloklar dışında metin şeklinde de yazılabilmektedir. Bunun için kod bölümü açıldıktan sonra Bloklar seçeneği yerine Metin seçeneği seçilmelidir. Aşağıdaki şekilde gösterilmiştir.

Avrupa Birliği tarafından ortak finanse edilmektedir



Bloklar ile kodlama

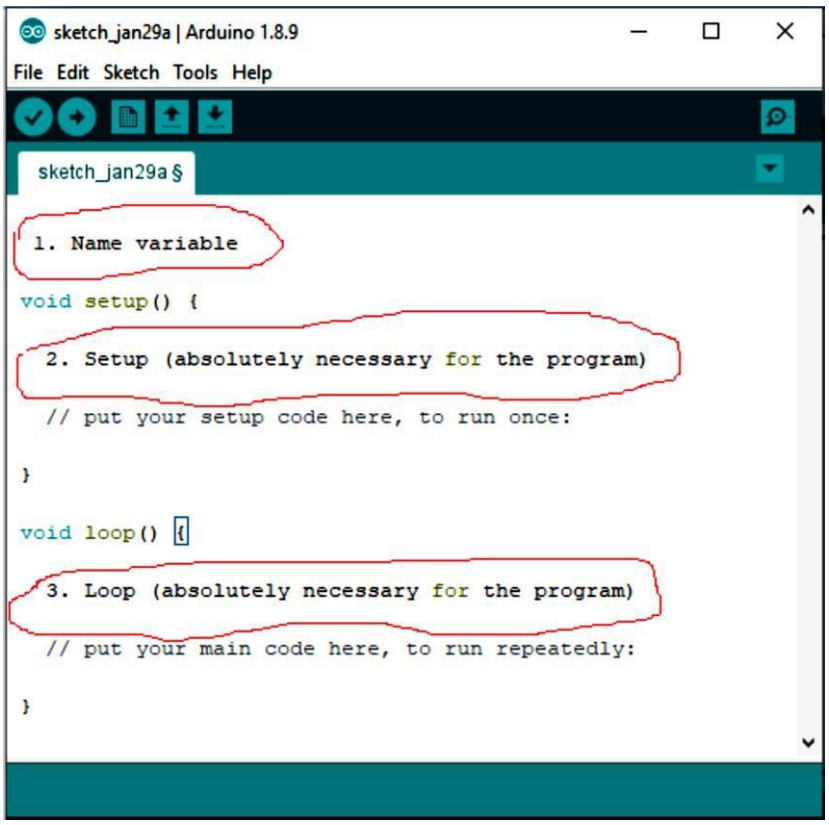


Metin ile kodlama

Arduino Programlamaya Giriş

Geliştirme ortamı ile Arduino programları yazılıp, derlenerek kartlar üzerine yüklenebilir. Arduino programlamada C / C++ /Java temelli bir dil yapısı kullanılır. Kütüphaneler sayesinde donanım seviyesine inmeye gerek yoktur, kod yazımında küçük büyük harf hassasiyeti vardır.

Arduino programı “sketch” olarak adlandırılır. Bir Sketch üç bölüme ayrılabilir.



Avrupa Birliği tarafından ortak finanse edilmektedir

1. Name variable(Değişken tanımlama-zorunlu değildir):

İlk bölümde programın elemanları isimlendirilmiştir. Bu kısım zorunlu değildir.

2. Setup(Kurulum-program için kesinlikle gereklidir):

Setup'daki kodlar sadece bir kez çalıştırılır.
Giriş ve çıkış pinleri belirlenmelidir.
Çıkış pinleri: Pin bir voltaj vermelidir. Örneğin: LED 'in yanması
Giriş pinleri: Kart bir voltaj okumalıdır. Örneğin: Bir anahtarın harekete geçirilmesi

3. Loop(Döngü-program için kesinlikle gereklidir):

Bu döngü kısmı kart tarafından sürekli olarak tekrarlanacaktır. Proje çalışır, sona gelince başa döner ve böyle devam eder.

Kodlama ile İlgili Bazı Kurallar

- Komutlar yan yana aynı satıra yazılabileceği gibi alt alta da yazılabilir. Ancak programın anlaşılabilirliği açısından alt alta yazmak faydalı olacaktır.
- Komutların sonuna (;) noktalı virgül konulur.
- Programın başında tanımlamalar kısmında kullanılacak kütüphaneler varsa #include komutu ile programa dâhil edilir.
- Türkçe karakter kullanılmamalıdır. Fakat açıklama satırları içerisinde (derleme işlemine dâhil edilmediğinden) kullanılabilir.
- Kod yazımında küçük büyük harf hassasiyeti vardır.

Noktalı virgül ";"

; Noktalı virgül bir komutu bitirmek için kullanılır. Bir komutu noktalı virgülle bitirmeyi unutmak derleyici hatasına neden olur.
Örnek: int a=13;

{ } (Süslü Parantez):

Süslü parantezinin ana kullanımları: Fonksiyonlar, Döngüler, Koşullu ifadelerdir.
Bir açılış süslü parantezini "{" her zaman bir kapanış süslü parantezi "}" izlemelidir.
Örnek:
void setup () **{** ……
…….
}

"//" simge:

"//" simgesi program sayfası içerisinde herhangi bir yere yazılabilir ve yazıldığı satır açıklama satırıdır. Açıklama satırları program yazarken hatırlatma amaçlı bazı notlar alınmasına yarar. Derleyici, "//" işaretinin olduğu satırı dikkate almaz.

Açıklama amacıyla kullanılan diğer bir yöntem ise “/*” ve “*/” işaretleridir. Program içerisinde istenilen yere “/*” işareti koyulduktan sonra gerekli notlar ya da açıklamalar yazılır. Sonrasında “*/” işareti konularak açıklama satırları bitirilir. Derleyici bu işaretler arasına yazılan hiçbir yazıyı dikkate almaz, sadece kullanıcıya ait satırlardır. Birden fazla açıklama satırı yazmak için bu yöntem tercih edilir.

Örnek:

```
// Bu komut ile bir pine 5V enerji verilir.
/* Bu komut ile bir pine 5V enerji verilir. Daha sonra LOW ile 0V gönderilir. */
```

Arduino Uygulamaları ve Komutlar

Aşağıda Arduino uygulamaları ile birlikte Arduino komut satırları verilmiştir. Böylelikle kodların tanımlanması, kullanım şekilleri ve devre üzerindeki etkileri gösterilmiştir.

• Void setup ()

Bir taslak başladığında fonksiyon çağrılır. Değişkenleri, pin modlarını başlatmak, kütüphaneleri kullanmaya başlamak vb. için kullanılır. setup() Bu fonksiyon Arduino kartının her açılışında veya sıfırlanışında yalnızca bir kez çalışacaktır.

Örnek Kod

```
int buttonPin = 3;

    void setup() {
      Serial.begin(9600);
      pinMode(buttonPin, INPUT);
    }

    void loop() {
      // ...
     }
```

• Void loop ()

loop() (döngü) fonksiyonu tam olarak isminin söylediği şeyi yapar ve ardışık olarak döngüye girerek programınızın değişmesine ve yanıt vermesine olanak tanır.

Örnek Kod

```
int buttonPin = 3;

  // setup initializes serial and the button pin
  void setup() {
    Serial.begin(9600);
    pinMode(buttonPin, INPUT);
  }

  // loop checks the button pin each time,
   // and will send serial if it is pressed
   void loop() {
     if (digitalRead(buttonPin) == HIGH) {
       Serial.write('H');
     }
     else {
       Serial.write('L');
     }

     delay(1000);
   }
```

DigitalWrite:

–Bu komut ile pinlerimize 0 V ya da 5 V veya 3.3V güç veririz. Örnek olarak pinMode() ile pin OUTPUT olarak ayarlanmış ise;

```
• digitalWrite(pinNo, HIGH) ile pine 5V verilir.
• digitalWrite(pinNo, LOW) ile pine 0V verilir.
• digitalWrite(LED, HIGH) //Burada LED isimli değişkene güç verdik.
• digitalWrite(motor,LOW)//Burada Motor isimli değişkeni durdurduk.
```

DigitalRead:

–Gelen 0-5 V değerlerini okumamızı sağlar.

–Digitalolarak 5V=1'e0V ise = 0'a eşittir.

•Kullanımı - digitalRead(Değişken)

```
-DigitalRead(bluetooth); //Bluetooth'dan gelen verileri okur.
-DigitalRead(potansiyometre);//Potansiyometreden gelen verileri okur
```

Avrupa Birliği tarafından ortak finanse edilmektedir

AnalogWrite:

–Bu komutla 0 ve 5 V arası değerleri vermemizi sağlıyor. Gelen ve giden elektrik akımı her zaman 0-255 arasıdır. Bu komutun yararı ise bazı bileşenlerin hızıyla oynamamızı sağlıyor ya da konumlanmasını sağlıyor.

–Örneğin motorun yavaş, orta hızlı veya çok hızlı olarak sürülmesini sağlıyor. Ya da bir servo motorun alacağı konumu ayarlama işlemini gerçekleştirebiliyoruz.

•Kullanımı - AnalogWrite(değişken, LOW)

```
–AnalogWrite(motor,255) // Motora 5V verdik.
–AnalogWrite(motor,204) // Motora 4V verdik.
```

AnalogRead:

–Bu komut ise gelen 0-5V arasındaki değerleri okumamızı sağlar.

•Kullanımı - analogRead(değişken);

```
–analogRead(bluetooth); //Bluetooth'dan gelen verileri okur.
–analogRead(potansiyometre);//potansiyometreden gelen verileri okur.
```

pinMode:

pinMode() fonksiyonu, belirli bir pini giriş veya çıkış olarak davranacak şekilde yapılandırmak için kullanılır.

Kullanımı: pinMode(pin, mod)

Parametreler: pin: Arduino pin numarası. mod: INPUT, OUTPUT veya INPUT_PULLUP.

```
pinMode(3, OUTPUT); // dijital pin 3'ü çıkış olarak ayarlar
pinMode(8, INPUT); // dijital pin 8'i giriş olarak ayarlar
```

delay(süremilisaniye):

Verilen süre kadar programın beklemesini sağlar. Paranteze 1000 yazarak 1000 milisaniye yani 1 saniye gecikme sağlanır.

Örnekler:

```
delay(1000);
```

Avrupa Birliği tarafından ortak finanse edilmektedir

Serial.begin(haberleşme_hızı);

–Bu komut ile seri haberleşmeyi açıyoruz. Bu serial haberleşmeden Metin, Bilgi, Sayı Gönderilebilir ya da Alabiliriz. Genellikle haberleşme hızı 9600'dur.

•Kullanımı - Serial.begin(haberleşme_hızı);

```
–Serial.begin(9600);//9600 Baund hızında seri haberleşmeyi başlattık
```

Serial.print();

–Bu komut ile sensörlerden gelen verileri ekrana yazdırabiliriz.

•Kullanımı - Serial.print(değişken);

```
–Serial.print("Merhaba Dünya");
```

–Serial.print(potansiyometre); // Potansiyometre verilerini ekrana yan yana yazdırıyor.

–Serial.print**ln**(potansiyometre); // Potansiyometre verilerini ekrana alt alta yazdırıyor.

❖ **for Komutu:**

For ifadesi, süslü parantezleri içine alınmış bir ifade bloğunu tekrarlamak için kullanılır. Bir artış sayacı genellikle döngüyü artırmak ve sonlandırmak için kullanılır. **for** ifadesi, herhangi bir tekrarlayan işlem için kullanışlıdır ve genellikle veri/pin koleksiyonları üzerinde çalışmak için dizilerle birlikte kullanılır.

Kullanımı:

```
for (başlangıç değeri; koşul; artış miktarı) {
// yapılacaklar;
}
```

Parametreler:

Başlangıç: önce olur ve bir kez tam olarak olur.
Koşul: Döngü boyunca her seferinde koşul test edilir; doğruysa, deyim bloğu ve artış yürütülür, koşul yeniden test edilir. Koşul yanlış olduğunda döngü sona erer.
Artış: koşul doğru olduğu sürece döngü boyunca her seferinde yürütülür.

```
for(x=0;x<100;x=x+2)
{
Serial.println(x); //100'den küçük çift sayılar ekrana yazılacak
}
```

Avrupa Birliği tarafından ortak finanse edilmektedir

❖ **İf - else komutu**

- **if** deyimi bir koşulu kontrol eder ve koşul “doğru” ise kontrolündeki deyimi veya deyim kümesini yürütür. Koşul “doğru” değil ise **else** işlemi yürütülür.

Kullanımı:

if (koşul) {

// koşul doğru olduğunda yürütülecek işlemler

}

koşul: bir boolean ifadesi (yani, doğru veya yanlış olabilir).

Else {

// koşul doğru değilse yürütülecek işlemler

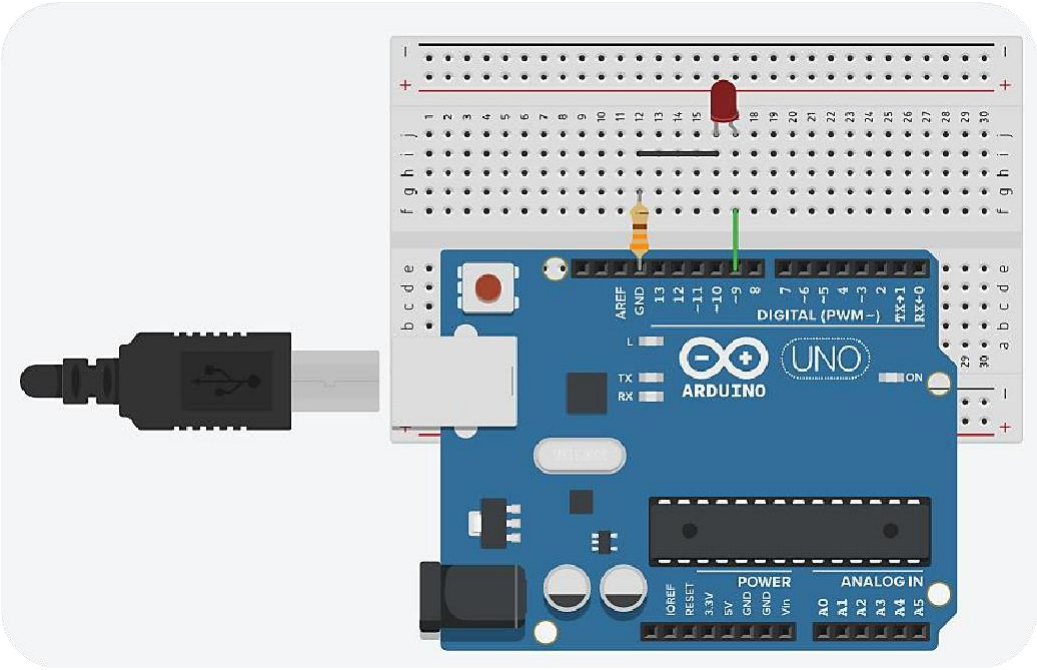
}

Not: İf komutu else olmadan tek başına da kullanılabilir.

Arduino Uygulamaları

❖ **Uygulama 1**

Dijital Çıkış (Led Yakma):
Arduino Uno ile 1 Ledi 1 saniye yakıp 1 saniye söndürme.



Malzemeler: 1 ad Arduino Uno R3, 1 ad Kırmızı LED ,1 ad 330 Ω direnç, Breadbord

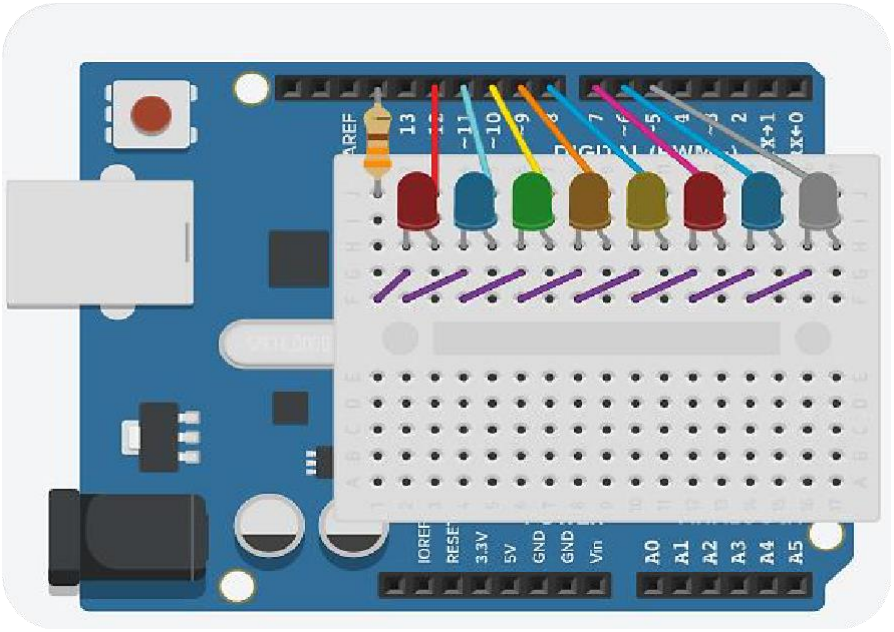
Led'in eksi (–) ucunu Arduino kartının GND pinine Ledin artı (+) ucunu direnç üzerinden Arduino kartının 9 numaralı dijital ucuna bağlanır. Direncin görevi; ledin üzerinden geçen akımı sınırlandırmak. Direnç kullanılmaz ise Led'in bozulması kaçınılmazdır.

Kod Satırı:

```
void setup()
{
  pinMode(9, OUTPUT); // 9 nolu pini çıkış olarak ayarladık
}

void loop()
{
  digitalWrite(9, HIGH); // 9 nolu pine 5V enerji verdik
  delay(1000); /* arduino yu bu satırda 1000 milisaniye (1 saniye)
                  bekletiyoruz */
  digitalWrite(9, LOW); // 9 nolu pinin 0V ile enerjisini kestik
  delay(1000); /* arduino yu bu satırda 1000 milisaniye (1 saniye)
                bekletiyoruz */
}
```

- **Uygulama 2**

Arduino ile 8 adet ledi sıra ile yakıp hepsinin birden söndürülmesi. Bu işlem sürekli devam eder.



Malzemeler: 1 ad Arduino Uno R3, 8 ad LED , 1 ad LED, 1 ad 330 Ω direnç, Breadbord

Ledlerin eksi (–) uçları birleştirilerek direnç üzerinden GND pinine bağlanır, Ledlerin artı uçları Arduino'nun sırasıyla 12-11-10-9-8-7-6-5-numaralı dijital pinlerine bağlanılır.

Kod Satırı:

```
void setup()
{
 pinMode(5, OUTPUT);
 pinMode(6, OUTPUT);
 pinMode(7, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(9, OUTPUT);
 pinMode(10, OUTPUT);
 pinMode(11, OUTPUT);
 pinMode(12, OUTPUT);
}
void loop()
{
 digitalWrite(5, LOW);
 digitalWrite(6, LOW);
 digitalWrite(7, LOW);
 digitalWrite(8, LOW);
 digitalWrite(9, LOW);
 digitalWrite(10, LOW);
 digitalWrite(11, LOW);
 digitalWrite(12, LOW);
 delay(1000);

 digitalWrite(5, HIGH);
 delay(1000);
 digitalWrite(6, HIGH);
 delay(1000);
 digitalWrite(7, HIGH);
 delay(1000);
 digitalWrite(8, HIGH);
 delay(1000);
 digitalWrite(9, HIGH);
 delay(1000);
 digitalWrite(10, HIGH);
 delay(1000);
 digitalWrite(11, HIGH);
 delay(1000);
 digitalWrite(12, HIGH);
 delay(1000);
}
```

❖ **Uygulama 3**

Uygulama 2 de ki program kodları, döngü kullanarak çok daha kısa yazabilir. Özellikle tekrarlama sayısı belli olan uygulamalarda **for** döngüsü tercih edilebilir.
For döngüsünü kullanarak uygulama 2'deki program aşağıdaki gibi kısaltılabilir.

Kod Satırı:

```
int i = 0;

void setup()
{
   for(i = 5;i<=12;i++){
   pinMode(i,OUTPUT);
   }
}

void loop()
{
   for(i = 5;i<=12;i++){
    digitalWrite(i,LOW);
   }
  delay(1000);

   for(i = 5;i<=12;i++){
    digitalWrite(i,HIGH);
    delay(1000);
   }
}
```

❖ **Uygulama 4**

Uygulama 2'deki devrede ledler sırasıyla sona kadar yanacak ve tekrar sondan başa doğru sırasıyla sönecek.

Kod Satırı:

```
int i = 0;

void setup()
{
  for(i = 5;i<=12;i++){
  pinMode(i,OUTPUT);
  }
}
```

```
void loop()
{
  for(i = 5;i<=12;i++)
  {
   digitalWrite(i,HIGH);
   delay(500);
  }
  for(i = 12;i>=5;i--)
  {
   digitalWrite(i,LOW);
   delay(500);
  }
}
```

- **Uygulama 5**

Uygulama 2'de ki devrede ledler dıştan içe doğru sırasıyla yanacak ve sonra tekrar içten dışa doğru sönecek.

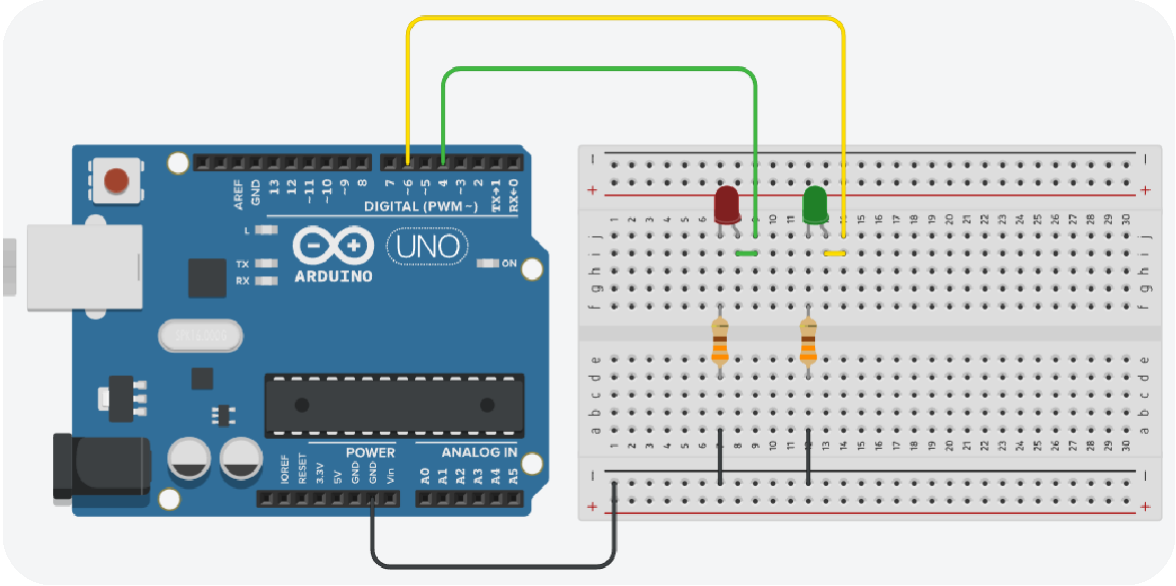
Kod Satırı:

```
int i = 0;

void setup()
{
  for(i = 5;i<=12;i++){
  pinMode(i,OUTPUT);
  }
}

void loop()
{
 for(i = 0;i<4;i++)
 {
 digitalWrite(5+i,HIGH);
 digitalWrite(12-i,HIGH);
 delay(500);
 }
 for(i = 3;i>=0;i--)
 {
 digitalWrite(5+i,LOW);
 digitalWrite(12-i,LOW);
 delay(500);
 }
}
```

❖ **Uygulama 6**

Flip-Flop:

Arduino ile 2 adet ledin sıra ile yakıp söndürülmesi. (Ledler aynı anda yanmayacak veya sönmeyecekler.) Gecikme 1 sn olacak.



Malzm: 1 ad Arduino Uno R3, 1 ad Kırmızı LED , 1 ad Yeşil LED, 2 ad 330 Ω direnç, Breadbord

Kod Satırı:

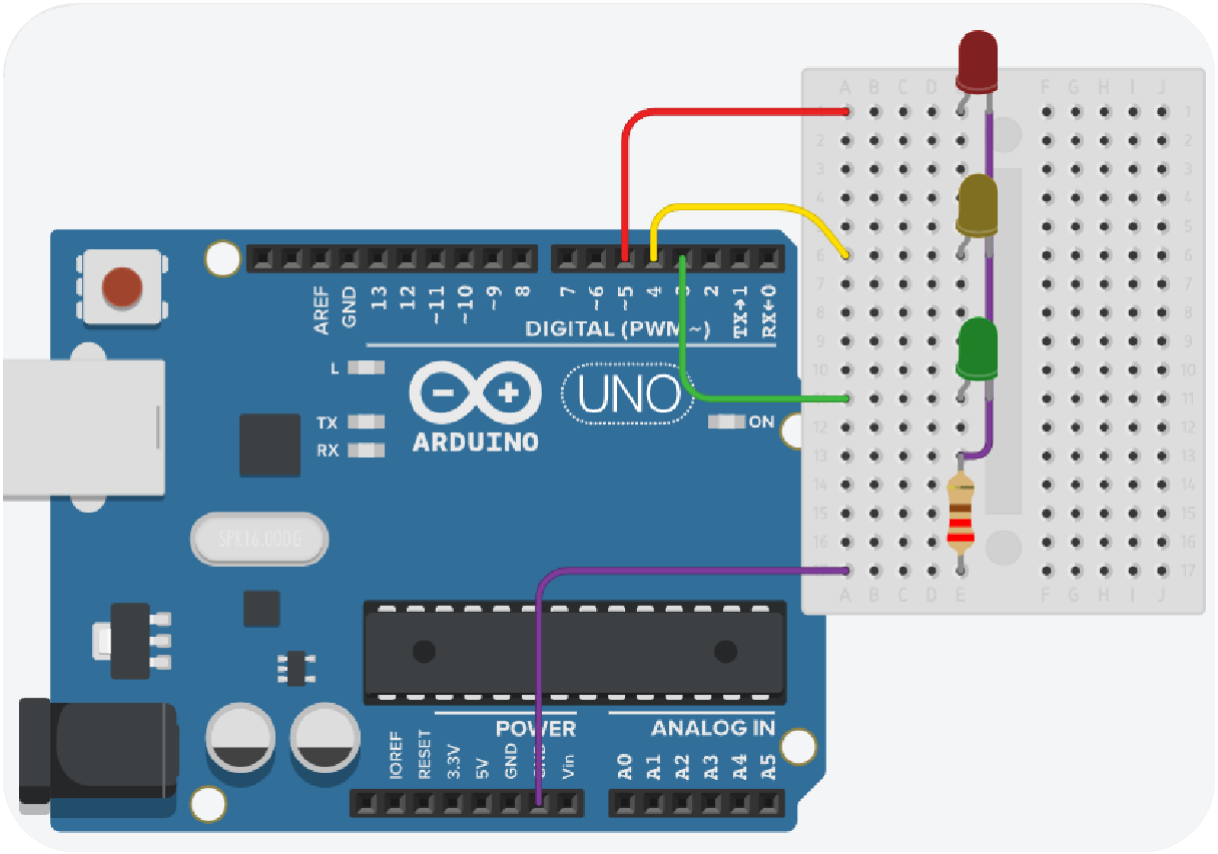
```
  /* Flip-Flop */

void setup() {

  pinMode(4, OUTPUT); // Kırmızı LED pin dijital çıkış olarak
ayarlandı
  pinMode(6, OUTPUT); // Yeşil LED pin dijital çıkış olarak
ayarlandı
}

  void loop(){

    digitalWrite(4, HIGH); // kırmızı LED'i yak
    digitalWrite(6, LOW); // yeşil LED'i söndür
    delay(1000); // programı 1000 milisaniye beklet

    digitalWrite(4, LOW); // kırmızı LED'i söndür
    digitalWrite(6, HIGH); // yeşil LED'i yak
    delay(1000); // programı 1000 milisaniye beklet
  }
```

Avrupa Birliği tarafından ortak finanse edilmektedir

❖ **Uygulama 7**

Trafik Lambası:

Devre Açıklaması: Bu projede, bir trafik ışığı sistemi kuracağız. Farklı renklerde (yeşil, sarı ve kırmızı) 3 adet led bulunmaktadır.



Malzemeler: 1 ad Arduino Uno R3, 1 ad Kırmızı LED , 1 ad Yeşil LED, 1 ad Sarı Led, 1 ad 220 Ω direnç, Mini Breadbord

Kod Satırı:

```
/* Trafik Lambası */

int kirmizi_LED = 5;
int sari_LED =4;
int yesil_LED = 3;


void setup() {  // burada pinlerimizi çıkış olarak ayarlıyoruz
  pinMode(kirmizi_LED, OUTPUT);
  pinMode(sari_LED, OUTPUT);
  pinMode(yesil_LED, OUTPUT);
}
```

Avrupa Birliği tarafından ortak finanse edilmektedir

```
  void loop() {
     digitalWrite(kirmizi_LED, HIGH); // kirmizi_LED'i 8 saniye yak
    digitalWrite(sari_LED, LOW);
    digitalWrite(yesil_LED, LOW);
    delay(8000);

     digitalWrite(kirmizi_LED, HIGH); // kirmizi_LED ve sari_LED'i 2
saniye yak
    digitalWrite(sari_LED, HIGH);
    digitalWrite(yesil_LED, LOW);
    delay(2000);

     digitalWrite(kirmizi_LED, LOW); // yesil_LED'i 5 saniye yak
    digitalWrite(sari_LED, LOW);
    digitalWrite(yesil_LED, HIGH);
    delay(5000);

     digitalWrite(kirmizi_LED, LOW); // tekrar, sari_LED'i 2 saniye
yak
    digitalWrite(sari_LED, HIGH);
    digitalWrite(yesil_LED, LOW);
    delay(2000);
}
```

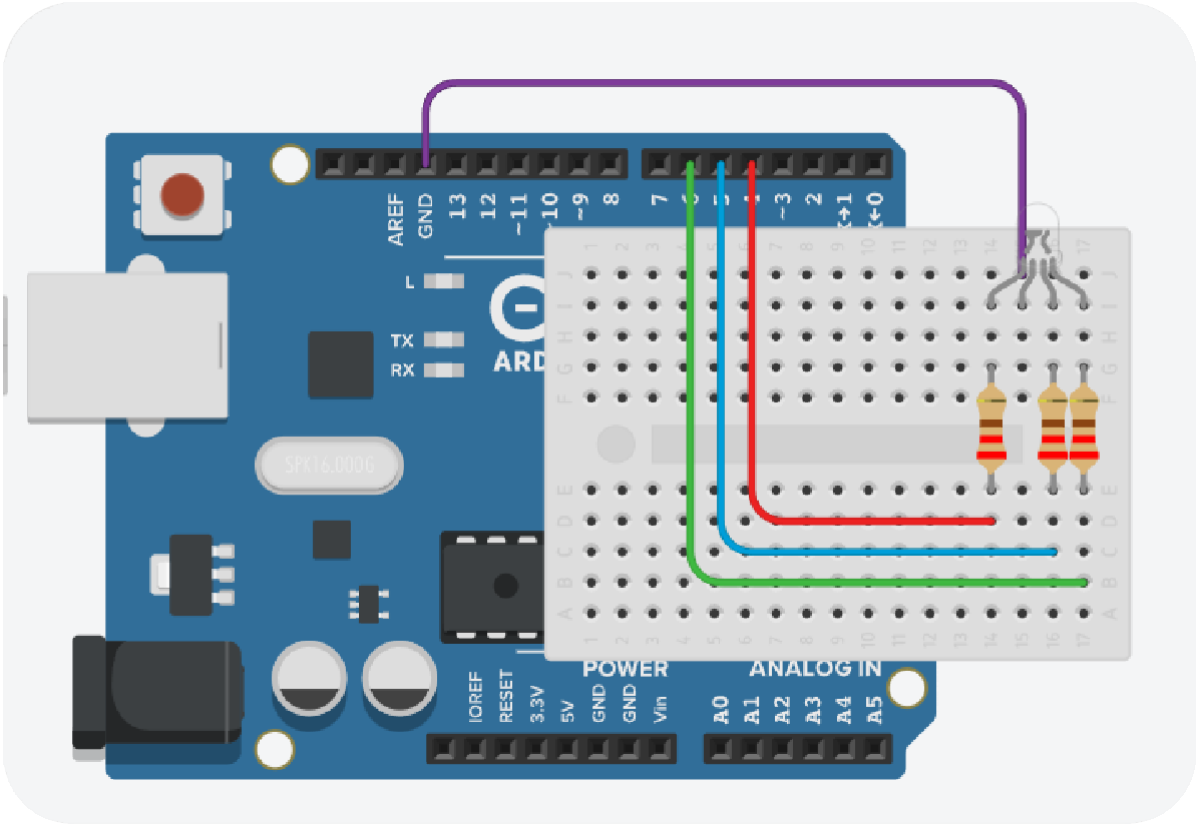
- **Uygulama 8**

RGB LED Uygulaması:

"RGB LED, tek bir kasaya yerleştirilmiş ortak olarak bağlı 3 LED'dir. Üç rengin ışık yoğunluğunu dijital olarak kontrol etmek mümkündür. Ayrıca PWM tekniği kullanılarak istenilen renkler elde edilebilir."

RGB LED'in her rengini 1 sn aralıklarla yakıp söndüreceğiz. Beyaz ışık için tüm ledleri aynı anda enerjilendirmemiz gerekiyor.

Malzemeler: 1 ad Arduino Uno R3, 1 ad RGB LED, 3 ad 220 Ω direnç, Mini Breadbord



Kod Satırı:

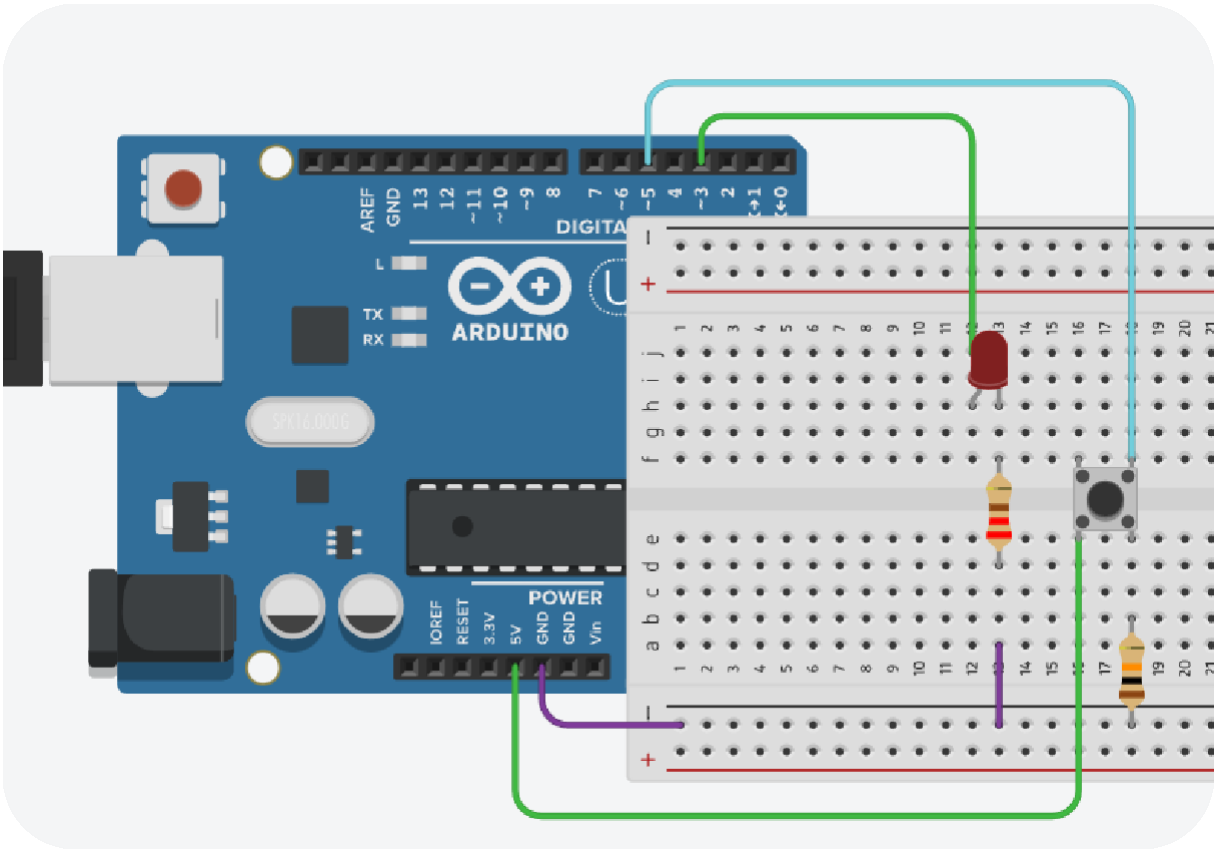
```
const int Kirmizi_Led=4; // Kirmizi_Led'e 4 değerini atadık
const int Mavi_Led=5; // Mavi_Led'e 5 değerini atadık
const int Yesil_Led=6; // Yesil_Led'e 6 değerini atadık

void setup()
{
  pinMode(Mavi_Led,OUTPUT);
  pinMode(Yesil_Led,OUTPUT);
  pinMode(Kirmizi_Led,OUTPUT);
}

void loop()
{
  digitalWrite(Mavi_Led, LOW); //Kirmizi_Led açık, diğerleri kapalı
  digitalWrite(Yesil_Led, LOW);
  digitalWrite(Kirmizi_Led, HIGH);
  delay(1000);

  digitalWrite(Mavi_Led, LOW ); // Yesil_Led açık.
  digitalWrite(Yesil_Led, HIGH);
  digitalWrite(Kirmizi_Led, LOW );
  delay(1000);
```

Avrupa Birliği tarafından ortak finanse edilmektedir

```
  digitalWrite(Mavi_Led, HIGH); // Mavi_Led açık.
  digitalWrite(Yesil_Led, LOW);
  digitalWrite(Kirmizi_Led, LOW);
  delay(1000);

// Üç ledi de aktif hale getirerek beyaz rengi açıyoruz.
  digitalWrite(Mavi_Led, HIGH);
  digitalWrite(Yesil_Led, HIGH);
  digitalWrite(Kirmizi_Led, HIGH);
  delay(1000);
}
```

❖ **Uygulama 9**

Bir butona bastığımızda Led yanacak, butonu bıraktığımızda Led sönecek.
Ledin eksi (–) ucu direnç üzerinden Arduino kartın GND pinine, ledin artı (+) ucu 3 numaralı pine, buton çıkışı 5 numaralı pine, 5 volt ve GND pinleri de butona bağlanır. Bu sayede butona basılmadığında 5 numaralı pine GND gelmesi sağlanır.



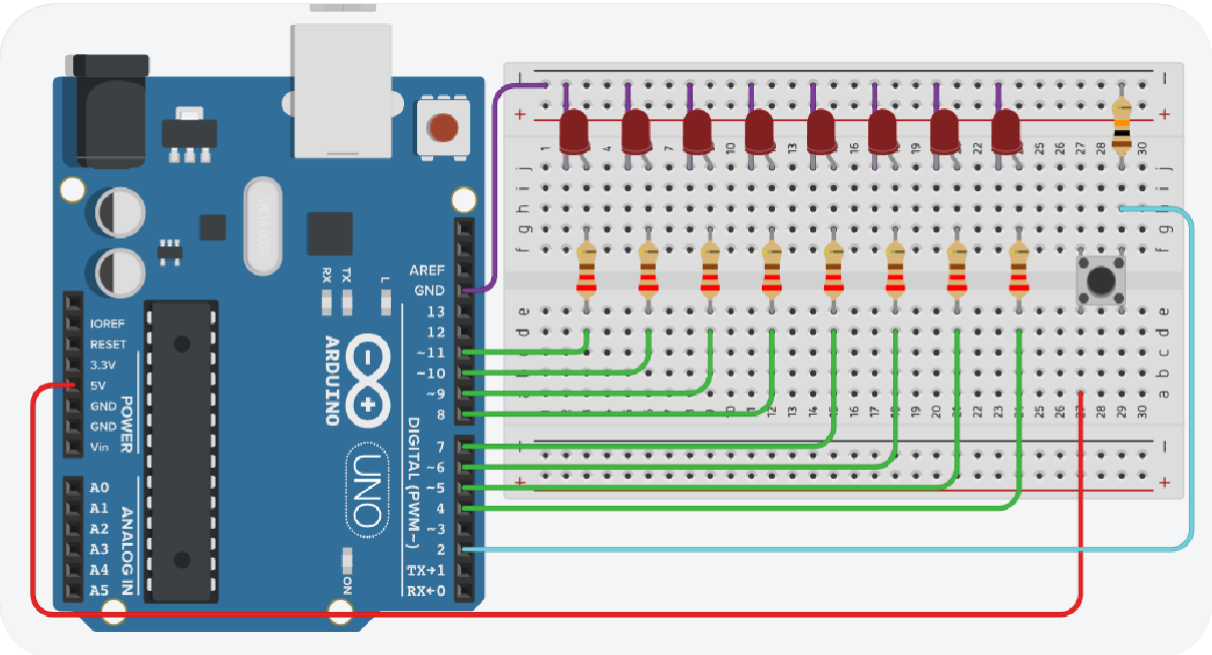
Malzemeler: 1 ad Arduino Uno R3, 1 ad LED, 1 ad 220 Ω direnç, 1 ad 10 kΩ direnç, Buton, Breadbord

Kod Satırı:

```
int buton=5;
int  Led=3;

void setup()
{
  pinMode(Led, OUTPUT);
  pinMode(buton, INPUT);
}

void loop()
{
   if (digitalRead(5)==HIGH){
     digitalWrite(Led,HIGH);
   }
   else {digitalWrite(Led,LOW);
   }
}
```

- **Uygulama 10**

Arduino'ya bağlı butona her basıldığında 8 adet Ledi sırasıyla yakan devre ve kodları.



Malzemeler: 1 ad Arduino Uno R3, 8 ad LED, 8 ad 220 Ω direnç, 1 ad 10 kΩ direnç, Buton, Breadbord

Ledlerin eksi (–) uçları birleştirilerek GND pinine bağlanır, Ledlerin artı uçları direnç üzerinden Arduinonun sırasıyla 11-10-9-8-7-6-5-4 numaralı dijital pinlerine bağlanılır. Buton çıkışı 2 numaralı pine, 5 volt ve GND pinleri de butona bağlanır.

Kod Satırı-1:

```
int led = 3;

void setup(){
  pinMode(2, INPUT);
  for(int i = 4;i<=11;i++)
  pinMode(i, OUTPUT);
}

void loop(){
  if(digitalRead(2)==1) {
   led++;
   delay(50); //Butonda oluşacak titreşimler engellemek için.
  }
 digitalWrite(led-1,LOW);
 digitalWrite(led,HIGH);

  if(led == 12) led = 4;
}
```

Not: Buton uygulamalarında butona basıldıktan sonra bir bekleme süresi eklenerek butonda oluşacak titreşimler engellenmiş olur. Titreşimleri engellemenin diğer yolu ise butona basıldıktan sonra işlem yapmayıp butonun bırakılmasının beklenmesidir. Bu işlem döngü kullanarak gerçekleştirebilir. Aşağıdaki kodlar bunu göstermektedir.

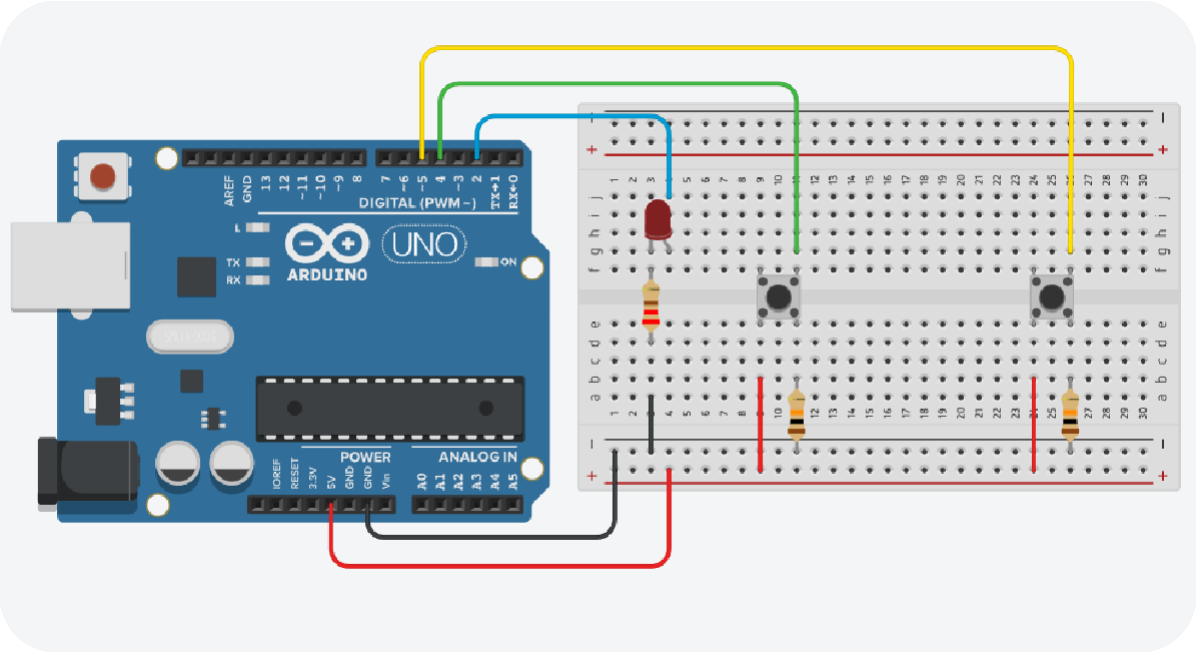
Kod Satırı-2:

```
int led = 3;

void setup() {
 pinMode(2, INPUT);
 for(int i = 4;i<=11;i++)
  pinMode(i, OUTPUT);
}

void loop() {
 if(digitalRead(2)==1)
  {
   while(digitalRead(2)==1); /*Bu komut ile el butondan çekilene
   kadar programın bu satırda beklemesini sağlar*/
   led++;
  }
 digitalWrite(led-1,LOW);
 digitalWrite(led,HIGH);

  if(led == 12) led = 4;
}
```

Avrupa Birliği tarafından ortak finanse edilmektedir

❖ **Uygulama 11**

LED’i iki butonla kontrol etme: Bir buton Led'i açar, diğer buton Led'i kapatır.



Malzemeler: 1 ad Arduino Uno R3, 1 ad LED, 1 ad 220 Ω direnç, 2 ad 10 kΩ direnç, 2 ad Buton, Breadbord

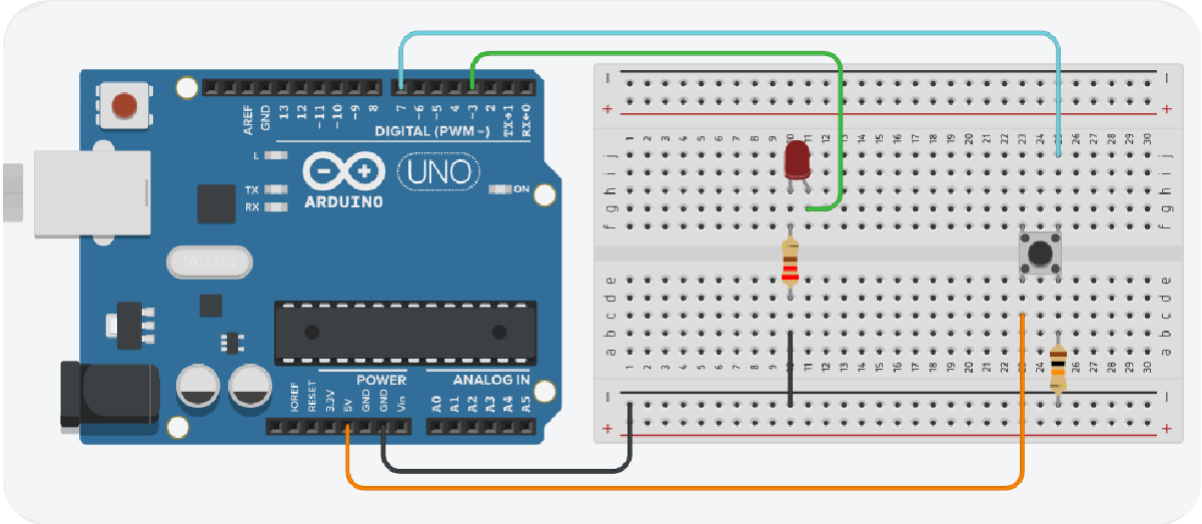
Kod Satırı:

```
int start=0;
int stop=0;

void setup(){
  pinMode(2, OUTPUT);
  pinMode(5, INPUT);
  pinMode(4, INPUT);
}

void loop(){
   start = digitalRead(5);//5 nolu pinden butona bak
   if (start == 1)//start eğer 1 ise
  {
    digitalWrite(2, HIGH); //2.pini yak
  }
   stop = digitalRead(4); //4 nolu pinden butona bak.
   if (stop == 1)//stop eğer 1 ise
  {
    digitalWrite(2, LOW);//2.pini söndür
  }
}
```

Avrupa Birliği tarafından ortak finanse edilmektedir

❖ **Uygulama 12**

LED'i tek butonla kontrol etme-1: Aynı butona basınca bir ledi sırayla açıp kapatır.



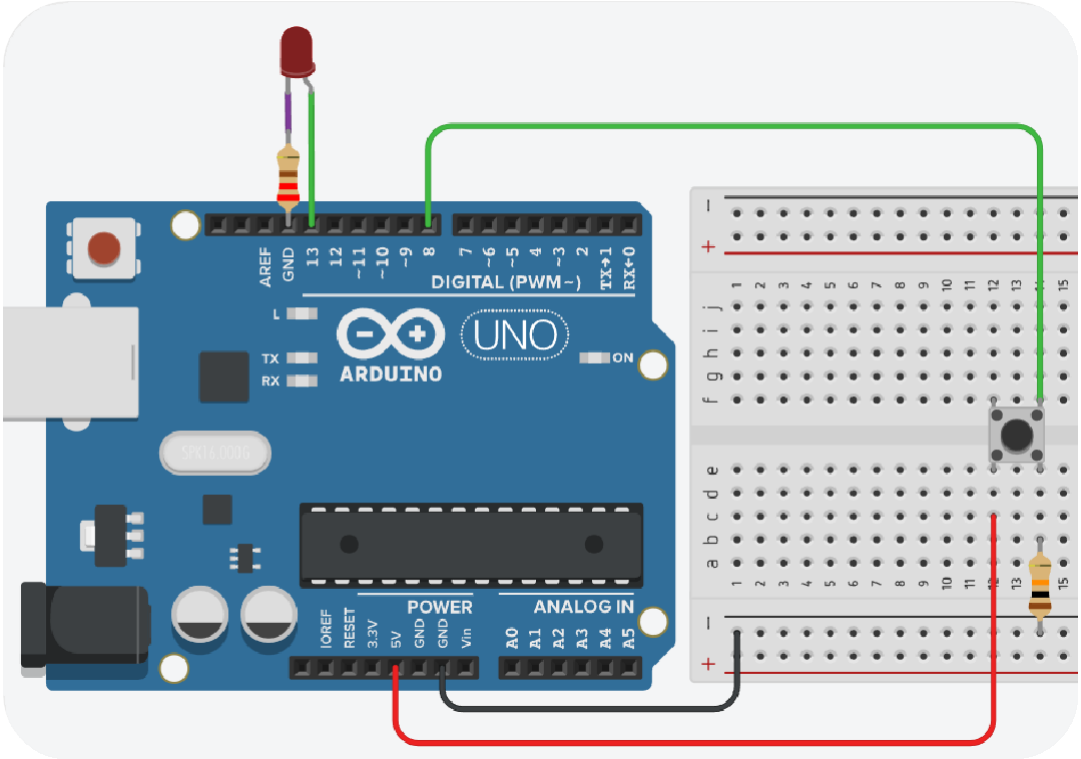
Malzemeler: 1 ad Arduino Uno R3, 1 ad LED, 1 ad 220 Ω direnç, 1 ad 10 kΩ direnç, 1 ad Buton, Breadbord

Kod Satırı-1:

```
int buton=0;

void setup(){
  pinMode(3, OUTPUT);
  pinMode(7, INPUT);
}

void loop(){
    while (buton==0){
     buton = digitalRead(7);
    }
  digitalWrite(3, HIGH);
    while (buton==1){
     buton = digitalRead(7);
    }
  buton=0;

    while(buton==0){
     buton = digitalRead(7);
    }
  digitalWrite(3, LOW);

    while (buton==1){
     buton = digitalRead(7);
    }
  buton=0;
}
```

Avrupa Birliği tarafından ortak finanse edilmektedir

LED’i tek butonla kontrol etme-2



Malzemeler: 1 ad Arduino Uno R3, 1 ad LED, 1 ad 220 Ω direnç, 1 ad 10 kΩ direnç, 1 ad Buton, Breadbord

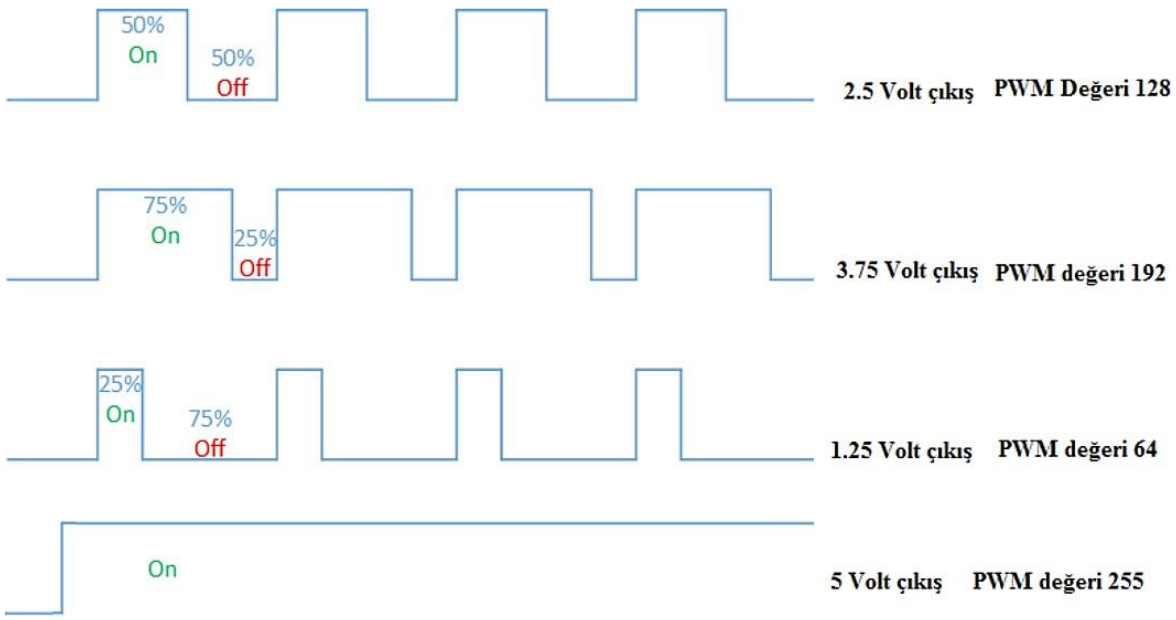
Kod Satırı-2:

```
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(8, INPUT);
}

void loop()
{
  while (digitalRead(8)==0)
  {}
  while (digitalRead(8)==1)
  {
    digitalWrite(13, HIGH);
  }


  while (digitalRead(8)==0)
  {}
  while (digitalRead(8)==1)
  {
    digitalWrite(13, LOW);
  }
}
```

Avrupa Birliği tarafından ortak finanse edilmektedir

Analog Çıkış RGB LED Uygulaması

Arduino Analog Çıkış: Arduino kartların üzerinde "~" işareti bulunan pinler PWM (Pulse Width Modulation) çıkışı verirler. PWM çıkışı çıkış voltajının belli bir süre 5 volt seviyesinde belli bir süre 0 Volt seviyesinde gönderilmesi ve bu işlemin çok hızlı bir şekilde sürekli yapılmasıdır. Aşağıda PWM sinyali görülmektedir.



PWM Çıkış Sinyalleri

Arduino UNO nun PWM çıkışı: 8 bittir. $2^8 = 256$ arduino ile çıkışa gönderebileceğimiz en büyük değerdir. Çıkıştan 5 Volt almak için 255, 0 Volt almak için 0 göndeririz. Bu değerlerden PWM çıkışının çözünürlüğünü tespit ederiz. 5 / 255 = 0.019 değerini elde ederiz. Bu Arduino Uno ile 0.019 Voltluk değişikliği algılayabileceğimiz anlamına gelir. Arduino Uno da kullanılan ATmega328P işlemcisi 6 adet PWM çıkışı vardır. Bu çıkışların frekansı işlemci içerisindeki zamanlayıcılar ile ayarlanabilir. Aşağıdaki tabloda hangi zamanlayıcının hangi pinler tarafından kullanıldığı görülmektedir.

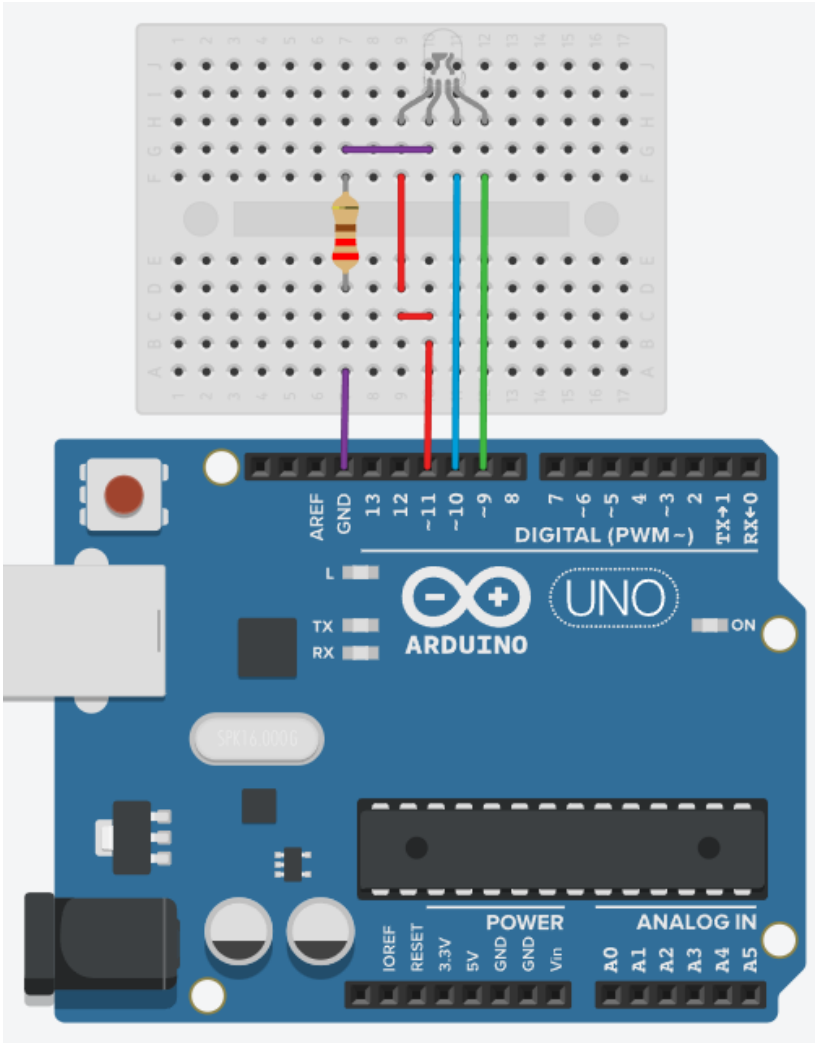
Timer 0	5 ve 6 numaralı pinler
Timer 1	9 ve 10 numaralı pinler
Timer 2	3 ve 11 numaralı pinler

❖ Uygulama 13

RGB ledi sırasıyla kırmızı, yeşil ve mavi olarak sırasıyla yakalım.

Arduino'nun PWM çıkışı veren 9,10 ve 11 numaralı pinleri RGB ledin kırmızı, yeşil ve mavi pinlerine bağlanır. GND pini de ledin eksi (–) ucuna bağlanır.

Malzemeler: 1 ad Arduino Uno R3, 1 ad RGB LED, 1 ad 220 Ω direnç, Mini Breadbord



Kod Satırı:

```
void setup()
{
 pinMode(9,OUTPUT);
 pinMode(10,OUTPUT);
 pinMode(11,OUTPUT);
}

void loop()
{
 analogWrite(9,255);
 analogWrite(10,0);
 analogWrite(11,0);
 delay(500);

 analogWrite(9,0);
 analogWrite(10,255);
 analogWrite(11,0);
delay(500);
```

Avrupa Birliği tarafından ortak finanse edilmektedir

```
 analogWrite(9,0);
 analogWrite(10,0);
 analogWrite(11,255);
 delay(500);
}
```

- **Uygulama 14**

RGB LED'in 1 saniye aralıklarla rastgele renklerde yakılması. Uygulama 11 devresine uygula.

Kod Satırı:

```
int kirmizi;
int yesil;
int mavi;

void setup()
{
 pinMode(9,OUTPUT);
 pinMode(10,OUTPUT);
 pinMode(11,OUTPUT);
 randomSeed(analogRead(A0));
}

void loop()
{
 kirmizi = random(255);
 yesil = random(255);
 mavi = random(255);
 analogWrite(9,yesil);
 analogWrite(10,mavi);
 analogWrite(11,kirmizi);
 delay(500);
}
```

Kullanılan komutlar;

randomSeed(analogRead(A0)); : Rastgele sayı üretecini başlatır. Bu üretecin üst üste aynı değerleri üretmemesi için boştaki bir analog girişe bağlarız. Boşta bırakılan analog girişler sürekli rastgele değerler üretir. Bu sayede random ile üretilen değerin sürekli farklı olması sağlanır.
kirmizi = random(255); : 0 ile 255 arasında rastgele bir değer üretir ve bu değeri kirmizi isimli değişkene aktarır.

Analog Giriş Potansiyemetre (Ayarlı Direnç) Uygulamaları;

Potansiyometre: Arduino Uno 10 bitli 6 adet analog girişe sahiptir. A0'dan A5'e kadar olan analog girişler aynı zamanda dijital olarak da kullanılabilir. Dijital olarak kullanılacağı zaman pinMode komutu ile tanımlanmalıdır. pinMode komutu ile tanımlanırken bu pinlere dijital pinlerde olduğu gibi numara verilmelidir. Bu pinlerin numaraları aşağıdaki gibidir.

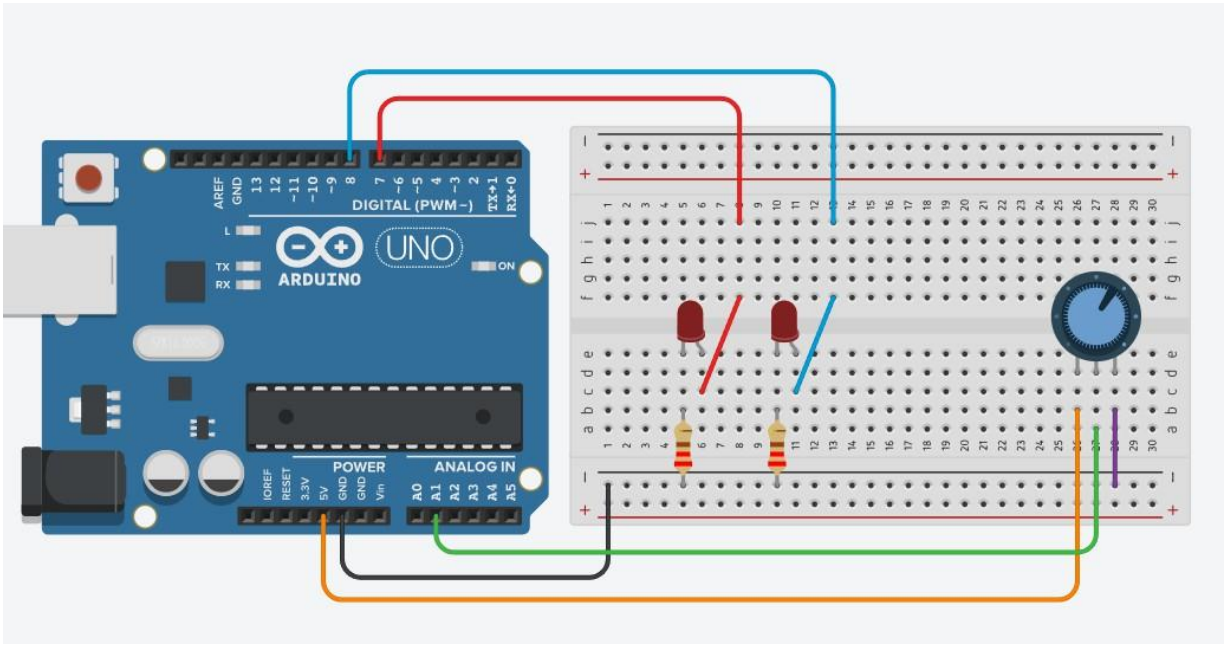
A0	14
A1	15
A2	16
A3	17
A4	18
A5	19

Analog girişler 10 bit olarak gösterilebildiği için 2^{10} = 1024 elde edilir. Arduino girişlerine 5 Volt uygulanabildiği için analog girişten 1024 okuduğumuzda 5 Volt uygulanmış demektir. Analog girişin hassasiyeti 5 / 1024 = 0.00488 Voltluk hassasiyet okuma yapabilir.

- **Uygulama 15**

A1 girişine bağlı potansiyometre 2.5 volt seviyesinin altında ise 7 numaralı çıkışa bağlı LED'in, 2.5 Voltun üstünde ise 8 numaralı çıkışa bağlı LED'in yakılması.

Potansiyometrenin orta ucu Arduino'nun analog giriş A1 pinine bağlanır. Potansiyometrenin diğer iki ucundan herhangi biri 5V pinine diğer ucu da GND pinine bağlanır. Ledlerin eksi (–) ucu direnç üzerinden GND ucuna, artı uçları da sırasıyla 7 ve 8 numaralı pinlere bağlanır.



Malzemeler: 1 ad Arduino Uno R3, 2 ad LED, 2 ad 220 Ω direnç, 1 ad Pot, Breadbord

Avrupa Birliği tarafından ortak finanse edilmektedir

Kod Satırı:

```
int pot;
int oran;

void setup()
{
 pinMode(7,OUTPUT);
 pinMode(8,OUTPUT);
}

void loop()
{
  pot = analogRead(A1);
  oran = map(pot,0,1024,0,255);
  if(pot >= 127)
  {
    digitalWrite(3,HIGH);
    digitalWrite(2,LOW);
  }
  if(pot < 127)
  {
    digitalWrite(2,HIGH);
    digitalWrite(3,LOW);
  }
}
```

Avrupa Birliği tarafından
ortak finanse edilmektedir