

**GELİŞMİŞ BUTON KONTROLÜ**

**16**

Önceki derslerimizde, ledin buton ile kontrolünü uygulamasını gerçekleştirmiştik. Ancak mikrodenetleyicinin çok hızlı çalışması ve butonda oluşan elektriksel arktan dolayı stabil çalışan bir uygulama değildi

**AMAÇ:**  Bu uygulamanın amacı, Buton yardımı ile bir ledin kontrolünü butona tek tıklama ile sağlayacağız. Böylece diğer sensör ve aygıtları kontrol ederken butona tıklandığında istenilen işlemin tam doğrulukta çalışmasını sağlayacağız.

**16.1 Buton ile Ledi yakıp söndürme**

Bir Ledin buton yardımıyla istenildiği gibi yakıp söndürülmesi doğru gerçekleştirilebildiğinde güzel bir örnektir. Çünkü bir Ledin yakılıp söndürülmesi farklı cihazların açılıp kapatılması ile aynı işleme tabidir. Bir ledi yakıp söndürme komutu başka bir şeyi açıp kapatmak içinde kullanılabilecektir. Örneğin bluetooth veya wifi yardımı ile bir şeyi kontrol etmek için buton yardımı ile kontrol işlemi için aynı kodlar kullanılabilecektir.

### **16.2 Kesme veya Buton durumunu kontrol etme**

Ledi yakmanın birçok yolu var. Tek butonla, çift butonla veya tuş takımı ile kontrol sağlanabilir. Ancak bunlarında ötesinde kullanılacak kodların yöntem ve farklılıkları vardır. Genellikle bir buton ile kontrol işlemi sağlanırken Kesme veya butonun durumunu kontrol etme yöntemi kullanılmaktadır.

Buton Durumunu Kontrol etme

Bu yöntemde digitalRead komutu ile kontrol işlemi sağlanır. Loop() içerisinde her daim butonun bağlı olduğu pinin dijital durumu hakkında bilgi alınır. Bu yöntemde butonun durumu hakkında bilgiye ancak kontrol ettiğimizde ulaşıyoruz. Bu durum; bize gelecek bir postanın ulaşıp ulaşmadığını kontrol etmek için, kapıdaki posta kutusunu sıklıkla kontrol etmemize benzetilebilir.

Kesme (Interrupt)

Adından da anlaşılacağı gibi kesmeler, mevcut işlemin kesintiye uğradığı ve yeni bir işlemin gerçekleştirilmesini sağlar. Bu yöntemde bir pinin devamlı kontrol ettirilmesine gerek yoktur. Arduino Kesme (interrupt) için belirlenen pinin dijital durumuna direk tepki göstererek çalışır. Bu durum; evde DVD’den film seyrederken, zilin çalmasına benzetilebilir. DVD durdurulur. Sonra kapı kontrol edilir. Daha sonra DVD filmi seyredilmesine devam edilir. Kapı zili olduğu için kapıyı kontrol etmeye gerek yoktur.

**Açıklama-1:**  Bu uygulamaları yapmadan önce 8.Ders Buton Kullanımı konusunu incelemeniz önerilir. Özellikle PullUp ve PullDown direnç mantığını tekrar etmeniz gerekecektir. Ayrıca, Arduino’nun dahili PullUp direncini ve kullanımını da araştırmalısınız.

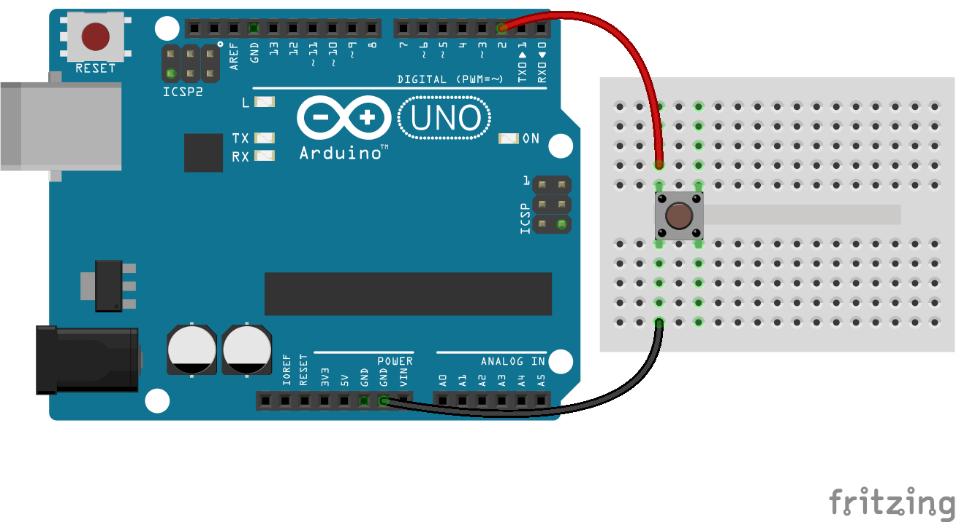
**Açıklama-2:**  Uygulamalarda dahili PullUp direnci kullanılmıştır. Led olarak da Arduino üzerindeki 13 nolu pine bağlı led üzerinden çalışılmıştır.

### **16.3 Buton Durumunu Kontrol Etme Tekniği**

Daha önce uyguladığımız yöntem, ilgili pinin durumu sürekli kontrol edilir ve kod pinin dijital değerine göre ne yapacağını belirler.

Uygulama 1: Bas Yak; Kaldır Söndür

Basitçe; butona basıldığında led yanacak, butondan el kaldırıldığında led sönecektir. Dahili PullUp direnç kullanıldığı için 2 nolu pin ilk durumda HIGH konumundadır. Butona basıldığında 2 nolu pin LOW konumuna gelecektir. (Bakınız: 8.ders).



int sayac=0; int LED = 13; int buton = 2;

void setup()

{

Serial.begin(9600);

Serial.print("Dosya: "); Serial.println(\_\_FILE\_\_);

Serial.print("Yüklendi: "); Serial.println(\_\_DATE\_\_);

Serial.println(" ");

pinMode(LED, OUTPUT);

digitalWrite(LED,LOW);

pinMode(buton, INPUT\_PULLUP);

}

void loop()

{

if ( digitalRead(buton) == LOW)

{

digitalWrite(LED, HIGH);

sayac++;

Serial.println(sayac);

}

else

{

digitalWrite(LED, LOW);

}

}

Uygulamayı çalıştırdığımızda Seri Port Ekranını da açalım. Butona basıldığında sayac değeri çok hızlı değişir. Bu da bize Arduino üzerindeki mikrodenetleyicinin ne kadar hızlı çalıştığını gösterir. Arduino kodu sürekli butonun durumunu kontrol eder (Else satırı buton basılı olmadığı sürece aktif olacaktır). Sürekli pinleri dinleme yöntemi yukarıdaki gibi basit örneklerde iyi çalışmaktadır. Ancak karmaşık uygulamalarda ise sorunlara yol açabilmektedir.

Uygulama 2: Bas Yak; Kaldır Söndür (Eski durumu hafızada tutarak)

Uygulama 1 ‘deki devre aynen kullanılacaktır. Bu yöntemde buton durumunu kaydeden iki değişken eklenir. Daha sonra bu iki değişkenler karşılaştırılır. Değişiklik varsa LED’i yakar veya kapatırız. Bu gereksiz yere LED pinini ayarlamak için zaman kaybedilmemesini sağlar.

int sayac=0; int LED = 13; int buton = 2;

// buton durumu eskiButonDurumu ve yeniButonDurumu

boolean eskiButonDurumu = LOW;

boolean yeniButonDurumu = LOW;

void setup()

{

Serial.begin(9600);

Serial.print("Dosya: "); Serial.println(\_\_FILE\_\_);

Serial.print("Yüklendi: "); Serial.println(\_\_DATE\_\_);

Serial.println(" ");

pinMode(LED, OUTPUT);

digitalWrite(LED,LOW);

pinMode(buton, INPUT\_PULLUP);

}

void loop()

{

yeniButonDurumu = digitalRead(buton);

if ( yeniButonDurumu != eskiButonDurumu )

{

if ( yeniButonDurumu == LOW )

{

digitalWrite(LED, HIGH);

sayac++;

Serial.println(sayac);

}

else

{

digitalWrite(LED, LOW);

}

eskiButonDurumu = yeniButonDurumu;

}

}

Uygulama 2’yi çalıştırdığımızda şeklen ilk uygulamadan farklı değildir. Aynı şekilde çalışmaktadır. Butona bastığımda LED yanar, elimi kaldırdığımda ise LED söner. Ancak fark Seri Port Ekranı açtığımızda daha net olarak anlaşılacaktır. Butona basıldığında sayac değeri bir artar (elektriksel arktan dolayı bazen birkaç sayı artabilir). LED’i tetikleme sadece butonda değişiklik olduğunda gerçekleşmektedir.

Tabiiki böyle bir uyguma için Arduino’ya gerek yoktur. Butona Led’i seri bağladığımız basit bir devre ile de bu uygulamayı gerçekleştirebiliriz ☺.

Uygulama 3: Geçiş düğmesi ile

LED'i yakmak için bir kez basmak ve LED’i söndürmek için tekrar basmak istiyorsak ne yapılmalıdır? Bunu yapmak için, butona basıldığında LED'in önceki durumunu bilmeliyiz. Bunu için LedDurumu adında bir değişken eklendi. LedDurumu kapalı ise Led sönecek. LedDurumu açık ise Led yanacaktır.

int sayac=0,LED = 13,buton = 2;

boolean eskiButonDurumu = LOW;

boolean yeniButonDurumu = LOW;

boolean ledDurumu = LOW;

void setup()

{

Serial.begin(9600);

Serial.print("Dosya: "); Serial.println(\_\_FILE\_\_);

Serial.print("Yüklendi: "); Serial.println(\_\_DATE\_\_);

Serial.println(" ");

pinMode(LED, OUTPUT);

digitalWrite(LED,LOW);

pinMode(buton, INPUT\_PULLUP);

}

void loop()

{

yeniButonDurumu = digitalRead(buton);

if ( yeniButonDurumu != eskiButonDurumu )

{

if ( yeniButonDurumu == LOW )

{ sayac++;

if(ledDurumu==LOW)

{

digitalWrite(LED, HIGH);

ledDurumu=HIGH;

Serial.print(sayac);Serial.println("- Led yandı");

}

else

{

digitalWrite(LED, LOW);

ledDurumu=LOW;

Serial.print(sayac);Serial.println("- Led söndü");

}

}

eskiButonDurumu = yeniButonDurumu;

}

}

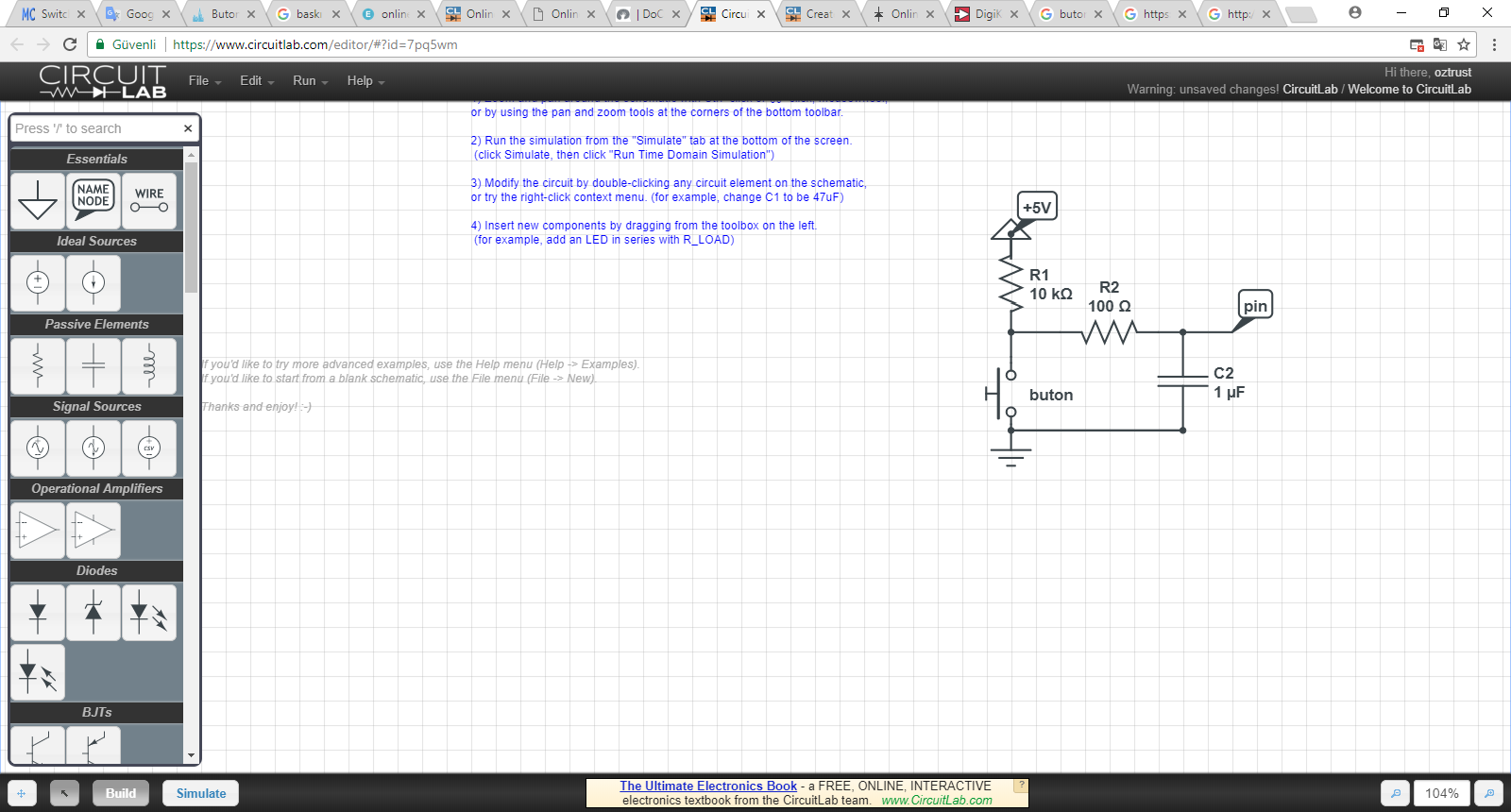
LedDurumu'un değerinin, LED pinine atanan değerle aynı olduğunu fark edebilirsiniz. ledDurumu değişkenini ayarladığımız için digitalWrite () ifadesini bir kez kullanabileceğimiz anlamına gelir.

if(ledDurumu==LOW) { ledDurumu=HIGH; Serial.print(sayac); Serial.println("- Led yandı"); }

else { ledDurumu=LOW; Serial.print(sayac);Serial.println("- Led söndü"); }

digitalWrite(LED, ledDurumu);

**Dikkat:** Uygulamada güvenilirlik sorunu var. Butona bastığımızda ledin bazen yanarken hızlı sönüp yandığı, sönükken hızlı yanıp söndüğü olacaktır. Bunun nedeni butondan kaynaklanan elektriksel arktır.

Açık bir butonun kapalı konuma getirilmesi  
işlemi esnasında elektriksel ark oluşur.   
Yani çok kısa bir zaman dilimi içeresinden   
buton kapanır, açılır, tekrar kapanır,   
tekrar açılır. Bu geçiş dönemi buton  
tamamen kapatılana kadar sürer.

Buton arkının çözümü yazılımsal ve   
donanımsal türde olmak üzere iki   
farklı şekilde yapılabilmektedir.

Donanımsal söndürmede ise ek olarak   
bir sönümleyici devresi gerekmektedir.   
Kondansatör ve dirençlerden oluşan   
sönümleyici devre yandaki şekildeki  
 gibi olmalıdır.

Yazılımsal ark söndürmede farklı yöntemler kullanılabilir. Lojik 1 durumunu algılayan kod satırından sonra ark süresi kadar gecikme tanımlamak gerekmektedir. Böylelikle program lojik durumunu tekrar kontrol edene kadar ark sönecektir.

Yazılımsal ark söndürme için aşağıdaki örneği tercih edebilirsiniz. Bu uygulamada, Buton durumunun 3 farklı örneklemesi alınır. Örneklemeler arasına kısa bir an beklenir. Örneklemeler birbirine eşit ise temiz bir durum gerçekleşmiştir.

yeniButonDurumu1 = digitalRead(buton);

delay(1);

yeniButonDurumu2= digitalRead(buton);

delay(1);

yeniButonDurumu3 = digitalRead(buton);

Programın son hali aşağıdaki gibi olacaktır.

int sayac=0,LED = 13,buton = 2;

boolean eskiButonDurumu = LOW;

boolean yeniButonDurumu1 = LOW;

boolean yeniButonDurumu2 = LOW;

boolean yeniButonDurumu3 = LOW;

boolean ledDurumu = LOW;

void setup()

{

Serial.begin(9600);

Serial.print("Dosya: "); Serial.println(\_\_FILE\_\_);

Serial.print("Yüklendi: "); Serial.println(\_\_DATE\_\_);

Serial.println(" ");

pinMode(LED, OUTPUT);

digitalWrite(LED,LOW);

pinMode(buton, INPUT\_PULLUP);

}

void loop()

{

yeniButonDurumu1 = digitalRead(buton);

delay(1);

yeniButonDurumu2= digitalRead(buton);

delay(1);

yeniButonDurumu3 = digitalRead(buton);

if ( (yeniButonDurumu1==yeniButonDurumu2) && (yeniButonDurumu1==yeniButonDurumu3))

{

if(yeniButonDurumu1 != eskiButonDurumu)

{

if ( yeniButonDurumu1 == LOW )

{

sayac++;

if(ledDurumu==LOW) { ledDurumu=HIGH; Serial.print(sayac); Serial.println("- Led yandı"); }

else { ledDurumu=LOW; Serial.print(sayac);Serial.println("- Led söndü"); }

digitalWrite(LED, ledDurumu);

}

eskiButonDurumu = yeniButonDurumu1;

}

}

}

Eklenen kodlarla birlikte Programın çalışması daha güvenilir hale gelmiştir.

### **16.4 Kesme (Interrupt) Tekniği**

Arduino gibi tek işlemciye sahip mikrodenetleyiciler çalışmasını işlem sırasına göre gerçekleştirirler. Normal şartlar altında bu işlem sırası asla terk edilmez. Ancak büyük ve karmaşık uygulamalarda buton gibi ilgili pinin okunma sırasını beklemesi sistemin çalışmasına olumsuz etkileri olabilir. Böyle durumlarda yapılan işlemin geçici durdurulması, daha önemli işlemlerin gerçekleştirildikten sonra tekrar kaldığı yerden çalışması için Kesmeler kullanılır.

Yukarıda kesme çeşitlerinden dış kesmenin üzerinde durmaya çalıştım. Dış kesmelerin yanı sıra zamanlayıcı(timer) kesmesi, haberleşme kesmesi vb kesme çeşitleri de bulunmaktadır. Zamanlayıcı kesmeleri, ayarlanan zaman gelince program akışı sırası terk edilip zamanlayıcı vektöründe belirlenen işlemleri yapılması sağlanır. Haberleşme kesmeleri’ de dış kesmenin haberleşme donanımlarında bulunan modelidir. Örneğin seri porttan veri gelmesi durumunda yapılmasını istediğimiz işlemler varsa bu kesmeler sayesinde gerçekleştirilebilir.

Biz bu konuda dış kesmeler anlatılacaktır. Arduino UNO’da dış kesme kullanımı için özelleşmiş iki adet pin vardır. Bu pinleri kullanarak sadece bir ayar yardımı ile dış kesme rahatlıkla kullanabilirsiniz.

|  |  |
| --- | --- |
| Board | Kesme için kullanılabilecek dijital pinler |
| Uno, Nano, Mini, other 328-based | 2, 3 |
| Mega, Mega2560, MegaADK | 2, 3, 18, 19, 20, 21 |
| Micro, Leonardo, other 32u4-based | 0, 1, 2, 3, 7 |
| Zero | 4. hariç tüm dijital pinler |
| MKR1000 Rev.1 | 0, 1, 4, 5, 6, 7, 8, 9, A1, A2 |
| Due | Tüm dijital pinler |

Pin Değişimine Göre Kesmeler

Pin değişimine göre dış kesme 4 farklı şekilde yapılabilir. Bunlar;

|  |  |
| --- | --- |
| Board | Kesme için kullanılabilecek dijital pinler |
| LOW | Pin gerilimi Lojik-0 olması durumudur. |
| CHANGE | Pin geriliminin değişmesi durumudur. (0’dan 5V veya 5V’tan 0’a gelindiğinde.) |
| RISING | Pin geriliminin Lojik-0′ dan Lojik-1′ e geçmesi durumudur. |
| FALLING | Pin geriliminin Lojik-1′ den Lojik-0′ a düşmesi durumudur. |

**Not:**  Burada bahsedilen Lojik-1, Uno, Mega vb. 5V mikrodenetleyiciler için 2.5V üzerisidir. Due gibi 3.3V mikrodenetleyiciler için 1.65V üzerisidir.

“attachInterrupt” Komutu

Kesme işlemi için “attachInterrupt” fonksiyonu kullanılır. 3 parametreden oluşur. İlk parametre pini; ikinci parametre kesme fonksiyonunun ismini ve son parametre kesme modunu belirler.

attachInterrupt( digitalPinToInterrupt(pinNo), fonksiyonAdi, kesmeModu )

**Örnek Kulanım**

void setup() {

attachInterrupt(digitalPinToInterrupt(2), fonksiyonAdi, kesmeModu);

}

void fonksiyonAdi()

{

}

“attachInterrupt” fonksiyonunda kullanılan digitalPinToInterrupt fonskiyonu kullanılacak dijital pin için interrupt numarasını döndüren bir fonksiyondur. Yani 2 nolu pin için 0, 3 nolu pin için 1 değeri döndürür. Kısaca şöyle kullanılabilir;

attachInterrupt(0, fonksiyonAdi, kesmeModu);

Fonksiyonumuzu aşağıdaki gibi oluşturabiliriz.

void fonksiyonAdi()

{

if (ledDurumu == HIGH) { ledDurumu = LOW; } else { ledDurumu = HIGH; }

digitalWrite(buton, ledDurumu);

}

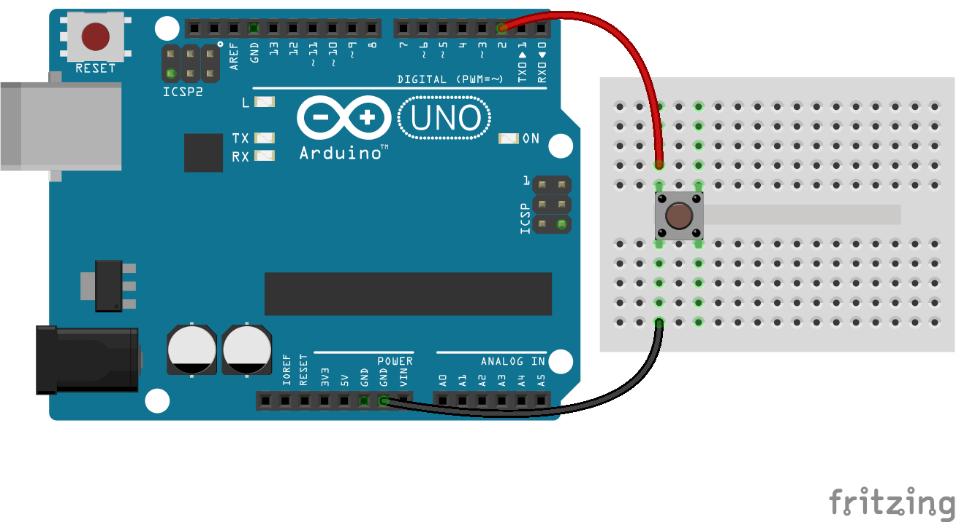
“detachInterrupt” Komutu

Kesmenin modunu değiştirmek veya tamamen devre dışı bırakmak için kullanılır.

detachInterrupt(digitalPinToInterrupt(2));

Uygulama 4: Buton ile Interrupt

Uygulama 3’de yapılan işlemi Interrupt kullanarak nasıl yapılır? Bu uygulamadaki fark buton pinini interrupt ile kontrol edilmesidir.



int sayac=0,LED = 13,buton = 2;

boolean ledDurumu = LOW;

void setup()

{

Serial.begin(9600);

Serial.print("Dosya: "); Serial.println(\_\_FILE\_\_);

Serial.print("Yüklendi: "); Serial.println(\_\_DATE\_\_);

Serial.println(" ");

pinMode(LED, OUTPUT);

digitalWrite(LED,LOW);

pinMode(buton, INPUT\_PULLUP);

attachInterrupt( 0, kesme, RISING );

}

void loop()

{

}

void kesme()

{

İf(ledDurumu==LOW){ ledDurumu=HIGH;}

else{ledDurumu==LOW;}

digitalWrite(LED,ledDurumu);

}

Butona basıldığında kesme fonksiyonu çağrılır. ledDurumu değişkeninin değeri değişir. Ardından Led’in durumu güncellenir. loop() fonksiyonunun içi boştur. Zira kodlar fonksiyon yardımı ile çağrılmaktadır. Butondaki arktan dolayı projenin güvenirliğinin tam olmamaktadır. ( Seri Port ekranından da kontrol edilebilir).

Uygulama 4: Kesme İşlemini Güvenilir Hale Getirme

Kesme işlemi uygulanırken butona basıldığını bir flag (bayrak) ile öğreniriz. Buton durumunu değişken ile kontrol etmek güvenilir değildir. Çünkü program kesme işleminden ana koda tekrar döndüğünde eski değer hala kullanılabilir. Bunu önlemek için **volatile** tip tanımlaması kullanılır.

Volatile tip tanımlayıcısı, basitçe bir değişken tanımının başında bulunup, bu değişkenin, program dışında bir etki altında bulunabileceğini ifade eder. Bu durumda derleyici volatile tip değişkenlerde her zaman son değeri kullanır.

int sayac=0,LED = 13,buton = 2;

boolean ledDurumu = LOW;

volatile boolean basildi=false;

unsigned long yeniTusSuresi=0;

unsigned long eskiTusSuresi=0;

unsigned int arkSuresi=200;

void setup()

{

Serial.begin(9600);

Serial.print("Dosya: "); Serial.println(\_\_FILE\_\_);

Serial.print("Yüklendi: "); Serial.println(\_\_DATE\_\_);

Serial.println(" ");

pinMode(LED, OUTPUT);

digitalWrite(LED,LOW);

pinMode(buton, INPUT\_PULLUP);

attachInterrupt( digitalPinToInterrupt(2), tiklama, RISING );

}

void loop()

{

if(basildi)

{

basildi=false;

yeniTusSuresi=millis();

if(yeniTusSuresi-eskiTusSuresi>=arkSuresi)

{

kesme();

}

eskiTusSuresi=yeniTusSuresi;

}

}

void tiklama()

{

basildi=true;

}

void kesme()

{

sayac++;

if(ledDurumu==LOW){ ledDurumu=HIGH;Serial.print(sayac); Serial.println("- Led yandı");}

else{ledDurumu=LOW;Serial.print(sayac);Serial.println("- Led söndü");}

digitalWrite(LED,ledDurumu);

}

Düğme düğmesine basıldığında, keyIsPressed () işlevini çağıran kesme tetiklenir. Tüm keyIsPressed () işlevi, keyPress değişkenini true değerine eşit olarak ayarlar.

Ana döngü () işlevinde, anahtarın değerini kontrol ederiz ve eğer doğruysa, tuşa basıldığını biliyoruz.

Geçiş zıplamasından dolayı çok hızlı bir şekilde tuşlara basabiliriz, bunu denemek ve filtrelemek için, çizimin son tuşa basılmasından bu yana ne kadar süre kaldığını ve zamanın geri dönüş süresi içinde kalması halinde tuşa basılmadığını görürsünüz. Düğme anahtarı için yaklaşık 10ms kullanıyorum. Bu farklı anahtarlar için farklı olacaktır. Ancak bu mükemmel bir çözüm değil.

Son not. Muhtemelen bu şekilde kesintiler kullanmam, daha sonra bunu mümkün olduğunu göstermek için rehbere dahil oldum ama temiz bir tetiğe sahip olmanızı sağlamak için yapmanız gereken tüm ekstra ayrıştırma işlerini sevmiyorum.

**ARAŞTIRMA:**  Adınızın harflerini 7 segment display yardımı ile 1 saniye ara ile yazdıran programı kodlayınız.

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………………………………………………

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ÖĞRENCİNİN** | | **DEĞERLENDİRME** | | | | | | |
| Adı | : | **Bilgi (20)** | **Düzen (20)** | **İş Sırası (20)** | **Sonuç (20)** | **Süre (20)** | **Toplam** | |
| Soyadı | : | **Rakam** | **Yazı** |
| Sınıfı | : |  |  |  |  |  |  |  |
| No | : | **Öğretmenin Adı:** | | | | | **İmza:** | |