

İÇİNDEKİLER

İÇİNDEKİLER.....	1
ATIF	2
AMAÇ	3
ARAÇLAR.....	3
ARAÇLARIN İNCELENMESİ	3
<i>Arduino UNO / Raspberry Pi Pico</i>	3
<i>Arduino IDE Serial Monitör</i>	3
<i>Rotary Encoder (with Push Button)</i>	3
<i>Arduino UNO IDE Kodlama Ortamı</i>	4
SİSTEMATİK ÇÖZÜMLEME	4
Proje Amacı Bağlamında Interrupt-Rotary Encoder Entegrasyonunu	6
Rotary Encoder Çalışma Mantığı ve Projedeki Vazifesi	6
A- Dönüş Hareketinin (Tick) Kullanımı:.....	6
B- Push Button Kullanımı	7
C- Rotary Encoder Pinlerine Dair Kodlar	7
Encoding ve Bt_Pushed ISR Fonksiyonları	8
A- Encoding().....	8
B- Bt_Pushed()	9
Diğer Fonksiyonlar.....	9
A- Gecikme()	9
B- IrqStart() ve IrqStop()	10
Main Fonksiyonu.....	10
KODLARIN TEK PARÇA HALİ	12

ATIF

<https://mekatronik.org/forum/threads/tuslu-rotary-encoder-moduel-yapimi.2652/page-16#302>

Sayın [@Gokrtl](#)'ın yoğun emeklerine dair bu başlıkta parça parça ürettiğim bazen neticesiz kalmış denemeler ve bunlara dair düşüncelerimi biraz daha elle tutulur hale getirmek üzere burada toparlamak istedim. Sayın [@Gokrtl](#)'a ve katkısı bulunan diğer forumdaşlara saygı ve teşekkürlerimle...

İstanbul - 12.04.2022

Hafy

AMAÇ

Proje amacımız, 2si tam ve 2si ondalık olmak üzere 4 basamaklı bir sayıyı üreterek Serial monitörde yazdırmak.

ARAÇLAR

- 1- Arduino UNO / Raspberry Pi Pico
- 2- Arduino IDE Serial Monitör
- 3- Rotary Encoder (with Push Button)
- 4- Arduino IDE Kodlama Ortamı

ARAÇLARIN İNCELENMESİ

Arduino UNO / Raspberry Pi Pico

C dili ile kodlanabilen ve Arduino IDE üzerinden programlanabilen, doğrudan ve kod vasıtasıyla kontrol edilebilen Giriş-Çıkış pinleri ve pek çok işlemsel özellik barındıran geliştirme kartlarıdır. Zaten tanınıyor ve yaygın biçimde kullanılıyor olduğu için burada detaylandırmaya ihtiyaç görülmemiştir.

Arduino IDE Serial Monitör

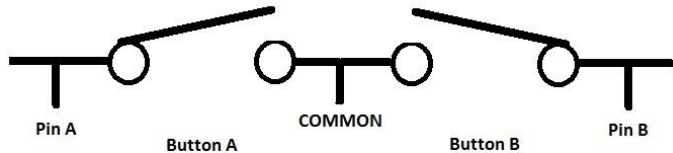
USB üzerinden bağlı bulunan geliştirme kartı ile yapılan serial haberleşmesinin kullanıcı arayüzü ile görüntülenmesini sağlayan araçtır.

Rotary Encoder (with Push Button)

Rotary Encoder, üzerinde bulunan milin kendi eksenini etrafında dönüşü ile bu milin temaslı (A ve B isimli) **iki butonun** aralarında 90° faz farkı bulunacak biçimde sıralı olarak tetiklenmesini sağlayan bir elektro-mekanik araçtır. Milin eksenini etrafında dönüşünde **tick** adını verdiğimiz hareket meydana gelir. Ayrıca milin eksenine dik olarak yerleştirilmiş diğer bir **push button** daha vardır.

a- Butonlar: A ve B butonu birer uçları birbirine ve Common isimli bir pine bağlıdır. A butonunun diğer ucu A Pini ve B butonunun diğer ucu B Pini olarak adlandırılır. Bu butonlar normal hallerinde açık-açık pozisyonundadır.

Common pini A ve B pinlerinin geliştirme kartı üzerinde bağlanacağı pinlerin Pull-up yahut Pull-down yapılmasına bağlı olarak ters polarlanır. Yani A-B pinleri +5/3.3volta (HIGH/1) çekilirse Common 0volta (LOW/0); A-B pinleri 0volta (LOW/0) çekilirse Common +5/3.3volta (HIGH/1) çekilir. Böylece pinlerin tetiklenmeleri esnasında geliştirme kartında bağlı bulundukları pinlerdeki lojik durum terslendiğinden Encoder'ın hareketi geliştirme kartı tarafından algılanır.



b- Tick: Rotary Encoderler, üzerlerindeki milin 360°lik bir turu farklı sayıdaki eşit açısal aralıklarda tamamlanacak şekilde üretilebilmektedir. Bu projede kullanılacak Encoder ise 360°lik bir turu 20 adımda tamamlayabilmektedir. Bu adımların her birisi "tick" olarak isimlendirilir. Her tick, ilk hallerinde açık-açık pozisyonunda bulunan A ve B

butonlarında sırasıyla kapalı-açık, kapalı-kapalı, açık-kapalı ve açık-açık durumlarını üreterek tamamlanır. Görüldüğü üzere bir tick 4 farklı durum üretmektedir. Demek ki Encoder'ımız 360°'lik bir turda toplam 80 farklı pin durumu üretebilmektedir. Bu da bir tick 18° ve her bir buton hareketi 4,5° açı bilgisi üretildiğini gösterir.

c- *Push Button*: Encoder üzerinde bulunan milin dikey ekseninde bastırılmasıyla temasa geçen diğer bir buton daha vardır. Bu buton ise bir pini geliştirme kartına bağlanarak Pull-up yahut Pull-down yapılır. Diğer pini ise bu duruma ters şekilde polarlanarak butonun tetiklenmesine bağlı olarak geliştirme kartını pini lojik olarak tersleneceğinden hareket algılanır.

d- *Rotary Encoder Modülü*: Bu modülün hazırlanmasında bilhassa A ve B pinlerinin bir kondansatör ile debounce edilmesi hızlı dönüşlerde sorun meydana getirmekte. Çünkü kondansatörler sıklaştıran palslerin arasında LOW-HIGH geçişini zamana yayarak olumsuz bir etki gösteriyor. Bunun kondansatörlerin t süresi üzerinden hesaplanmasıyla nasıl etkileri olduğu görülebilir. Nitekim Sayın @Gokrtl tarafından paylaşılan osiloskop incelemesine dair videonun 15. saniyesinden itibaren bu etkinin palsler arasındaki 90°'lik faz farkını etkilediği görülecektir. Ancak Buton pininin böyle bir kondansatörle debounce edilmesine ihtiyaç görülebilir. Diğer taraftan Modülde kullanılacak dirençler hakkında da bazı durumlara değeriendirmek icap ediyor. Modülde 10Kohm Pull-up direnci kullanıldığında Pico pinleri 50Kohm ile internal Pullup yapılırsa eşdeğer direncin $1/50 + 1/10 = 6/50 = 0,12$ Kohm'a karşılık geldiğini görürüz. Arduino UNO'nun internal pullup dirençlerinin 20Kohm ile 50Kohm arasında olduğu belirtiliyor. Bu durumda $1/20 + 1/10 = 3/20 = 0,15$ Kohm ile 0,12 Kohm eşdeğer direnç gibi davranacaktır. Bu kadar düşük pull-up direncinin modülün 1 tick hareketi esnasında meydana gelen gürültüye sebep olması mümkün gibi görünüyor.

Arduino UNO IDE Kodlama Ortamı

Bilindiği gibi bu ortam C ve C++ dilinde ve kendine özgü metotlarla üretilen kodun, bir derleyici vasıtasıyla makine diline çevirerek geliştirme kartına yüklenmesini sağlar.

SİSTEMATİK ÇÖZÜMLEME

Bilindiği gibi elektronik devreler harici bir müdahaleye ihtiyaç duymayacak biçimde tasarlandıklarında, içerdikleri komponentlerin yapılarına bağlı bir senkronizasyon ile çalışırlar. Projede kullandığımız Arduino UNO / Raspberry Pi Pico gibi geliştirme kartları ise temel komponent olan işlemci bakımından işlemlerini üretmek ve yürütebilmek adına diğer unsurlarla senkronizasyonu sağlamak üzere kendi saatlerini kullanırlar.

Ancak dışarıdan işlemcilerin bu zamanlamasına senkronize yanıt veremeyecek müdahaleler öngörülüyor ise işlemcinin kendi zamanlamasını bu müdahaleye uydurması sağlanmaktadır. Bu müdahaleler neler olabilir? Bir kullanıcın sisteme işleme ihtiyacı duyacağı veriler, işlemci tarafından kontrol edilen çeşitli elektronik ve mekanik araçların durumlarına dair veriler olabilir. Ancak SPI, I2C, UART, Serial gibi haberleşme biçimleri işlemcilerin iletişime geçtikleri diğer cihazlarla aralarında senkronizasyon temin ederek gerçekleştiğinden burada çözümlenmesi gereken konular değildir. Bu bahiste ana mesele şöylece şekillenmektedir:

1- Rotary Encoder ile kullanıcı tarafından üretilen hareketin işlemci tarafından doğru algılanması,

2- Bu hareketin sonucunun görüntülenmesi yoluyla kullanıcı tarafından doğruluğunun teyit edilebilmesi,

3- Hareketle üretilen veriler üzerinden amaçlanan işlemlerin gerçekleştirilmesi.

Bu üç meselenin temelinde yatan nokta şudur. Biz insanlar, saniyenin 1/6'sından daha kısa sürede meydana gelen olayları algılayamıyor ve tepki üretemiyoruz. Her ne kadar bazı ritmik durumlarda bu türden kısa süreleri yakalayabilsek de saniyenin 1/100'ünden 1/1000000'una kadar kısa zamanlarda işlem gerçekleştiren elektronik cihazlarla doğrudan iletişim kurabilmemiz imkânsız. Dolayısıyla bu cihazlar insanlar tarafından kullanıma elverişli arayüzlerle donatılır ve cihazların, bu arayüzler üzerinden aldıkları tepkilere zamansal olarak hizalanması temin edilir. Burada söz, insan-işlemci senkronizasyon vasıtası olan Hardware Interrupt'a gelmektedir. Ki bu vasıta işlemciler tarafından pek çok çeşit cihazın kontrolünde de kullanılmaktadır.

Esasında Arduino gibi geliştirme ortamlarında bir Interrupt'a bağlı olmadan da pek çok buton ve encoder uygulaması tasarlanıp çalıştırılabilmektedir. Öyleyse neden Interrupt gibi karmaşık olaylar silsilesi bir konuya girmek durumunda kalıyoruz? Şöyle izah edelim:

1- Ana kod bloğumuzu oluşturan Main Loop içerisinde bir butonun durumunu kontrol etmek için bazı yöntemler bulunmakta: Maksat bunlardan birisi olan IF-ELSE yapısıyla Main Loop'un her döngüsünde bir defa kontrol ederek; diğeri olan WHILE yapısıyla da butonda bir hareket meydana geldiğini kaçırmamak üzere Main Loop döngüsünü bekleterek gerçekleşir.

Aşağıda iki yöntem üzerinden örnek verilecek olup önce ikisinde de kullanılması muhtemel değişken tanımlamalarını yapalım:

```
int Button = 3;
int Led = 4;
void setup() {
  pinMode(Button, INPUT_PULLUP);
  pinMode(Led, OUTPUT);
  digitalWrite(Led, LOW);
}
```

a- IF-ELSE ile buton kontrolü: Burada Main Loop'a ait diğer kod örneklerine değinilmeyecektir.

```
void loop() {
  if (Button == LOW) {
    digitalWrite(Led, HIGH);
  }
  //Diğer kodlar
}
```

Burada örneği verildiği şekliyle butonun basılı olup-olmadığı, Main Loop'un her döngüsünde bir anlığına kontrol edilir. Main Loop'un her döngüsü ne kadar zamanda tamamlanırsa butondaki hareket de ancak o kadar sürede bir kontrol edilebilir. Yani Main Loop 1 saniye sürerse butona basıldığı saniyede 1 defa kontrol edilebilir. Bizim ise butona bastığımızda 1 saniye kaybetmek istemiyorsak Main Loop döngüsünün tam da bu kontrol koduna geldiği ânı yakalamamız gerekir.

b- WHILE ile gerçekleştirilen diğer çözüm, IF-ELSE bloğu ile yapılan kontrolünde temeli oluşturan "zamanını yakalama" problemini aşıyor gibi. Bu kod ise şöyledir:

```
void loop() {  
  WHILE (Button == HIGH) {  
    delay(100);  
  }  
  pinMode(Led, OUTPUT);  
  //Diğer Kodlar  
}
```

Burada görüleceği üzere butonun basıldığını anlamak üzere Main Loop döngüsü bekletilmektedir. Ancak buton basıldığında gerekli işlem yapıldıktan sonra Main Loop'un döngüsünü devam ettirmesine izin verilir.

Bu iki yöntemin de temelinde yatan problem bir TRIGGER problemidir. Yani bize butonun basıldığını takip ederek haber verecek bir tetikleyici eksikliğidir. Yazılım ve özellikle elektronik cihaz haberleşmelerine aşina olanlar TRIGGER kavramını da biliyorlardır. İşte Hardware Interrupt, bu türden bir TRIGGER görevi görmekte, programın ana döngüsündeki akışı herhangi bir aksamaya uğratmadan buton vesair olaylarda işlemci zamanlamasını bu olayların zamanına hizalamaktadır.

2- Interrupt karmaşasına girmek üzere ikinci nedenimiz ise, Main Loop gibi yoğun kodlar arasında yaşanacak aykırı durumlarda bizim buton kontrolümüze ne şekilde sıra geleceğinin çoğu kere öngörülebilir olmamasıdır. Oysa işleyişi doğru kavrandığında Hardware Interrupt, buton, Rotary Encoder vb. ihtiyaçların hemen hepsinde öngörülebilir ve sistem kaynaklarını doğru kullanabilen bir vasıta. Bu karmaşayı aşmanın yolu ise Interrupt işleyişi ile buna bağlı kullanılacak komponentlerin iyi kavranmasıdır.

Gelin Interrupt-Rotary Encoder entegrasyonunu bu bağlamda değerlendirelim.

Proje Amacı Bağlamında Interrupt-Rotary Encoder Entegrasyonunu

Rotary Encoder Çalışma Mantiğı ve Projedeki Vazifesi

A- Dönüş Hareketinin (Tick) Kullanımı:

Daha yukarıda anlattığımız gibi Rotary Encoder bir tick esnasında A ve B pinlerinin durumu itibarıyla şu çıktıları verir. A-B pinleri Pull-up yapıldığı ve Encoder Saat yönünde döndürüldüğünde:

Ân	A Pini (CLK)	B Pini (DATA)	BINARY
0	1	1	11
1	0	1	01
2	0	0	00
3	1	0	10

A-B pinleri Pull-up yapıldığı ve Encoder Saatin tersi yönde döndürüldüğünde:

Ân	A Pini (CLK)	B Pini (DATA)	BINARY
0	1	1	11
1	1	0	01
2	0	0	00

3	0	1	01
---	---	---	----

Görüleceği üzere bir tick sürecinde Encoder'dan 4 defa veri alınabilmekte ve buna binaen bir Tick'te 4 işlem yapma imkânı bulunmaktadır.

Proje amacı 2si tam ve 2si ondalık olmak üzere 4 basamaklı bir sayının her basamağını ayrı ayrı ayarlamak söz konusu olduğuna göre:

1- *Tamamlanması neredeyse 1 saniye dahi sürmeyen 1 tick ile basamak rakamlarında 4 artış gerçekleştirmek pek de kullanışlı değildir.*

2- *Saniyenin ¼'lük kısmında ürettiğimiz her bir rakamın ekranda yazdırılmasıyla kaybolması aynı kısa sürede gerçekleşeceğinden Encoder ile elde edilen etkinin gözlenmesi zorlaşacaktır.*

3- *Bir basamak değeri 3'e ayarlanacak olursa Encoder milinin, bir Tick'in ¾'lük kısmında sabitlenmesi gerekecektir. Ancak Encoder mili bir tick tamamlanacak şekilde tasarlanmış olduğundan bu da mekanik olarak imkân dışı bulunacaktır.*

4- *Diğer taraftan biz bu dört basamaklı sayıyı basamak basamak ayırmadan sürekli encoder çevirerek oluşturmak isteseydik, bir Tick'te 4 işlem yapma imkânı bize büyük katkı sunardı. 00.00-39.99 arasındaki 4000 farklı sayıyı bir turda 80 (1 tur 20 Tick, 1 Tick 4 defa işlem yapma) veri üreten encoder ile 50 turda oluşturabilirdik. Ancak yapmaya çalıştığımız şey her basamakta 0-9 arasında toplamda 10 farklı değer döndürmek olduğuna göre bu kadar baş döndürücü bir hıza zaten ihtiyaç da yok.*

Görüldüğü üzere hem döndürme hareketinde kendi kontrolümüzü sağlamak hem de Rotary Encoder üzerinden çalıştıracağımız Interrupt kesmelerini sağlıklı kullanabilmek önemlidir. Bu nedenle Interrupt Mode (yani tetiklenme biçimi) ayarında, bir Tick'te 1 rakam ilerleme çözünürlüğü sağlayacak tercihte bulunmamız gerekiyor. CLK görevi üstelenen A pinimizi geliştirme kartımızın 0 nolu Interrupt pini olan 2 nolu dijital pine bağlayacağız. Ve bu pini de Pull-up yapacağız.

Yukarıdaki iki tabloda vurgulandığı üzere, CLK pini FALLING hâlinde iken DATA pini 1 ise Saat Yönünde; DATA pini 0 ise Saatin Tersi Yönde bir dönme hareketi tespit edilmektedir. Bu nedenle CLK pininin FALLING hâlinde DATA pininin durumunun okunabilmesi gerektiğinden onu da geliştirme kartımızın 7 nolu dijital pinine bağlıyoruz.

Bu da A pininin 0'dan 1'e geçtiği 1. ânı bize gösteren FALLING hâlidir. A pini FALLING durumuna uğradığında çalışacak fonksiyonumuz ise Encoding() olacaktır.

B- Push Button Kullanımı

Rotary Encoder'ın dönüş hareketiyle basamaklara gelen sayıları değiştireceğiz. Ancak hangi basamakta işlem yaptığımızı ise Encoder'ın butonuna basarak seçeceğiz. Bu nedenle Push Button pinlerinden birisini geliştirmek kartımızın 1 nolu Interrupt pini olan 3 nolu dijital pine bağlayacağız. Bu pin Pull-up yapılacaktır. Diğeri ise GND'ye bağlanarak butona tıklanması durumunda bağlı bulunduğu Interrupt pinini terslemesi sağlanacaktır. Böylece buton hareketi algılanacaktır. Butona dair Interrupt ise Bt_Pushed fonksiyonunu çağıracaktır.

Buraya kadar elde ettiğimiz veriler üzerinden pin tanımlamaları ve Interrupt yapılandırmasına ait kodları şöylece düzenleyebiliriz:

C- Rotary Encoder Pinlerine Dair Kodlar

Encoder Pinlerine Dair Değişkenler

```
int RE_A = 2;  
int RE_Bt = 3;  
int RE_B = 7;
```

Dönüş yönü bilgisini taşıyacak olan değişken tanımlanıyor:

```
String Yon = "BOS";
```

Encoding() fonksiyonu içerisinde B pininin durumunu tutacak olan PinB değişkeni:

```
volatile bool PinB = 0;
```

Tick Gerçekleştiği Bilgisini taşıyacak değişken:

```
volatile bool ticked = 0;
```

Butona Basıldığı Bilgisini taşıyacak değişken:

```
volatile bool pushed = 0;
```

Buton ile işlenecek basamak verisine tutacak değişken tanımlanıyor:

```
volatile int BasamakNo = 0;
```

Üretilen sayıyı tutacak değişken tanımlanıyor:

```
float sayi = 0;
```

setup() fonksiyonu:

```
void setup() {
```

Encoder Pinlerinin yapılandırılması:

```
pinMode(RE_A, INPUT_PULLUP);  
pinMode(RE_Bt, INPUT_PULLUP);  
pinMode(RE_B, INPUT_PULLUP);
```

Interrupt Kesmelerinin yapılandırılması:

```
attachInterrupt(digitalPinToInterrupt(RE_A), Encoding, FALLING);  
attachInterrupt(digitalPinToInterrupt(RE_Bt), BT_Pushed, FALLING);
```

Serial Monitör kurulumu:

```
Serial.begin(115200);  
}
```

Encoding ve Bt_Pushed ISR Fonksiyonları

A- Encoding()

Encoder mili dönme hareketine başladığı anda CLK pini LOW'a çekilecek ve Encoding() isimli fonksiyon çağırılacak. Bu nedenle Encoding() fonksiyonu yazılıyor:

```
void Encoding(){
```

RE_A şeklinde isimlendirilen CLK pininin FALLING hâlinde fonksiyon çağırıldığına göre RE_B ile isimlendirilen DATA (B) pininin durumunu kontrol ederek dönüş yönünü tespit edilecek bu nedenle önce bu pinin durumu okunarak PinB değişkenine atanıyor:

```
PinB = digitalRead(RE_B);
```

Interrupt kesmesi meydana geldiğinde işlemler tamamlanana kadar başka kesme meydana gelmemesi amacıyla tüm kesmeler deaktive ediliyor. Kesmeler, Interruptlar ile amaçlanan işlemler gerçekleştirildiğinde yeniden aktive edilecek:


```
IrqStop();
```

RE_A şeklinde isimlendirilen CLK pininin FALLING hâlinde fonksiyon çağırıldığına göre RE_B ile isimlendirilen DATA (B) pininin PinB değişkenine aktarılan durumu kontrol edilerek dönüş yönü tespit ediliyor:

```
if (PinB == 1) {Yon = "SAG";} {Yon = "SOL";}
```

Tick gerçekleştiği bilgisi ilgili değişkene işleniyor:

```
ticked = 1;
```

Debounce gecikmesi eklenerek ve Encoding() fonksiyonundan çıkılıyor:

```
Gecikme(50);  
}
```

B- Bt Pushed()

Encoder butonuna basılmasıyla RE_Bt pini LOW'a çekileceğinden Bt_Pushed() isimli fonksiyon çağırılacak. Bu nedenle Bt_Pushed() fonksiyonu yazılıyor.

```
void Bt_Pushed(){
```

Interrupt kesmesi meydana geldiğinde işlemler tamamlanana kadar başka kesme meydana gelmemesi amacıyla tüm kesmeler deaktive ediliyor. Kesmeler, Interruptlar ile amaçlanan işlemler gerçekleştirildiğinde yeniden aktive edilecek:

```
IrqStop();
```

Butona basıldığı bilgisi pushed değişkenine işleniyor:

```
pushed = 1;
```

Debounce gecikmesi eklenerek Bt_Pushed() fonksiyonundan çıkılıyor:

```
Gecikme(50);  
}
```

Diğer Fonksiyonlar

Burada Interruptlarla ilgili zaman gecikmelerine ihtiyaç duyulması hâlinde kullanılacak bir Gecikme() fonksiyonu tanımlanıyor. Buna gerek duyulmasının sebebi Interruptlar içerisinde delay(), millis(), micros() gibi bilinen diğer zaman fonksiyonlarının çalışmaması. Bu fonksiyonda Interrupt esnasında çalışabilen tek zaman fonksiyonu olan delayMicroseconds() kullanılacaktır.

Interrupt kesmelerinin aktive ve deaktive edilmesi birkaç yerde tekrar edeceğinden kod karmaşasına yol açmamak üzere bunların birer fonksiyonla çağırılması amaçlandı.

A- Gecikme()

Fonksiyon ile beraber içerisinde, çağırıldığı zaman ne kadar süre gecikme sağlanmasını hesaplamakta kullanılacak bir de değişken tanımlaması yapılıyor.

```
void Gecikme(volatile int Sure){
```

Gecikmeyi sağlayacak olan for döngüsü kurularak, delayMicroseconds() fonksiyonu ile 1 milisaniye karşılığı 1000 mikrosaniye gecikme ayarlanıyor. Bu sayede fonksiyon çağırıldığında belirtilecek Sure değişkeni kadar milisaniye gecikme temin edilecektir. Sonra da fonksiyondan çıkılıyor:

```
for (int a = 0; a < Sure; a++) {delayMicroseconds(1000);}
```

```
}
```

B- IrqStart() ve IrqStop()

IrqStart() fonksiyonu tanımlanıyor ve Interrupt kesmelerini yapılandıran ve aktive eden kodlar eklenerek fonksiyondan çıkılıyor:

```
void IrqStart(){
  attachInterrupt(digitalPinToInterrupt(RE_A), Encoding, FALLING);
  attachInterrupt(digitalPinToInterrupt(RE_Bt), BT_Pushed, FALLING);
}
```

IrqStop() fonksiyonu tanımlanıyor ve Interrupt kesmelerini deaktive eden kodlar eklenerek fonksiyondan çıkılıyor:

```
void IrqStop(){
  detachInterrupt(digitalPinToInterrupt(RE_A));
  detachInterrupt(digitalPinToInterrupt(RE_Bt));
}
```

Main Fonksiyonu

Bu kısımda artık buton ve tick yoluyla elde edilen veriler Serial monitöre yazdırılacak.

loop() fonksiyonu tanımlanıyor

```
void loop(){
```

Encoder ile Bt_Pushed kesmesi gerçekleştiği kontrol ediliyor gerçekleşmişse BasamakNo değişkeninin artırılması ve 0-3 arasında değer alması sağlanıyor:

```
if (pushed) {
  BasamakNo++;
  if (BasamakNo == 4) {BasamakNo = 0;}
  Serial.print(BasamakNo);
  Serial.println(". Basamak");
}
```

Gerekli işlemleri gerçekleştirildiği için buton basıldığı durumunu taşıyan değişken sıfırlanıyor:

```
pushed = 0;
```

Hangi kesme gerçekleştiğinin kontrolüne dair IF-ELSE bloğunun ikinci kısmına geçiliyor

```
} else if (ticked1) {
```

Tick yönü ve BasamakNo değerine göre ilgili basamak değeri işlenecek

```
if (Yon == "SAG"){
  switch (BasamakNo){
    case 0:
      sayi+=0.01;
      if (sayi > 30.00) {sayi = 00.00;}
      break;
    case 1:
      sayi+=0.1;
      if (sayi > 30.00) {sayi = 00.00;}
      break;
    case 2:
      sayi+=1;
      if (sayi > 30.00) {sayi = 00.00;}
      break;
    case 3:
```

```

    sayi+=10;
    if (sayi > 30.00) {sayi = 00.00;}
    break;
}
} else if (Yon == "SOL"){
    switch (BasamakNo){
    case 0:
        sayi-=0.01;
        if (sayi < 0) {sayi = 30.00;}
        break;
    case 1:
        sayi-=0.1;
        if (sayi < 0) {sayi = 30.00;}
        break;
    case 2:
        sayi-=1;
        if (sayi < 0) {sayi = 30.00;}
        break;
    case 3:
        sayi-=10;
        if (sayi < 0) {sayi = 30.00;}
        break;
    }
}
}

```

Gerekli işlemleri yapıldığı için tick gerçekleşme bilgisi ile yön bilgisini taşıyan değişkenler sıfırlanıyor, üretilen sayı Serial monitöre yazdırılıyor ve ilgili IF-ELSE kontrolünden çıkılıyor:

```

    ticked = 0;
    Yon = "BOS";
    Serial.print("Sayı: ");
    Serial.println(sayi);
}

```

Buton ve Tick kesmelerinin gerçekleşmesine dair işlemler tamamlandığı IF-ELSE ile kontrol edilerek IrqStart() fonksiyonu ile tüm kesmeler aktive ediliyor. Böylece Main Loop tamamlanmış oluyor.

```

    if (ticked + pushed == 0) {IrqStart();}
}

```

KODLARIN TEK PARÇA HALİ

```
int RE_A = 2;
int RE_Bt = 3;
int RE_B = 7;
String Yon = "BOS";
volatile bool PinB = 0;
volatile bool ticked = 0;
volatile bool pushed = 0;
volatile int BasamakNo = 0;
float sayi = 0;

void setup() {
  pinMode(RE_A, INPUT_PULLUP);
  pinMode(RE_Bt, INPUT_PULLUP);
  pinMode(RE_B, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(RE_A), Encoding, FALLING);
  attachInterrupt(digitalPinToInterrupt(RE_Bt), Bt_Pushed, FALLING);
  Serial.begin(115200);
}

void Encoding(){
  PinB = digitalRead(RE_B);
  IrqStop();
  if (PinB) {Yon = "SAG";} else {Yon = "SOL";}
  ticked = 1;
  Gecikme(50);
}

void Bt_Pushed(){
  IrqStop();
  pushed = 1;
  Gecikme(50);
}

void Gecikme(volatile int Sure){
  for (int a = 0; a < Sure; a++) {delayMicroseconds(1000);}
}

void IrqStart(){
  attachInterrupt(digitalPinToInterrupt(RE_A), Encoding, FALLING);
  attachInterrupt(digitalPinToInterrupt(RE_Bt), Bt_Pushed, FALLING);
}

void IrqStop(){
  detachInterrupt(digitalPinToInterrupt(RE_A));
  detachInterrupt(digitalPinToInterrupt(RE_Bt));
}

void loop(){
  if (pushed) {
    BasamakNo++;
    if (BasamakNo == 4) {BasamakNo = 0;}
    Serial.print(BasamakNo);
    Serial.println(". Basamak");
    pushed = 0;
  } else if (ticked) {
    if (Yon == "SAG"){
```

```

switch (BasamakNo){
case 0:
    sayi+=0.01;
    if (sayi > 30.00) {sayi = 00.00;}
    break;
case 1:
    sayi+=0.1;
    if (sayi > 30.00) {sayi = 00.00;}
    break;
case 2:
    sayi+=1;
    if (sayi > 30.00) {sayi = 00.00;}
    break;
case 3:
    sayi+=10;
    if (sayi > 30.00) {sayi = 00.00;}
    break;
}
} else if (Yon == "SOL"){
switch (BasamakNo){
case 0:
    sayi-=0.01;
    if (sayi < 0) {sayi = 30.00;}
    break;
case 1:
    sayi-=0.1;
    if (sayi < 0) {sayi = 30.00;}
    break;
case 2:
    sayi-=1;
    if (sayi < 0) {sayi = 30.00;}
    break;
case 3:
    sayi-=10;
    if (sayi < 0) {sayi = 30.00;}
    break;
}
}
}
ticked = 0;
Yon = "BOS";
Serial.print("Sayi: ");
Serial.println(sayi);
}
if ((ticked + pushed) == 0) {IrqStart();}
}

```