



Makaleler

---

# Hız Ayarlı Çoklu DC Motor Kontrolü

# Hız Ayarlı Çoklu DC Motor Kontrolü

*Yazan: Mustafa Tufaner, Düzenleyen: Canol Gökel - 18 Kasım 2006*

## Giriş

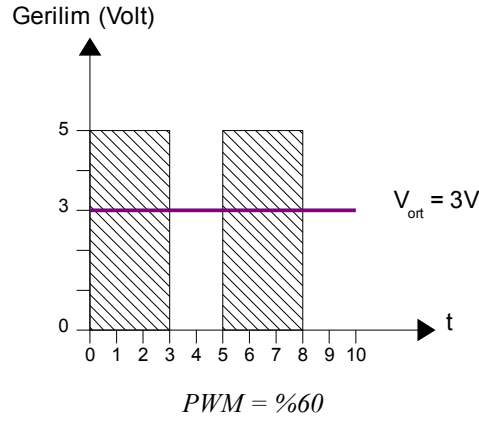
Robotikte sıkça kullanılabilecek bir uygulama ile karşınızdayız. Nasıl ki uygulamalarınızda robotunuzun beynine mikrodenetleyici yerleştiriyorsunuz, bu beynin eli ayağı olarak da tabii ki motorlar kullanacaksınız. Bu uygulama ile robotumuza, beyniyle elini nasıl kontrol edebileceğini öğreteceğiz. Tabii hangi hızda hareket ettireceğini dışarıdan gelen kullanıcı komutları belirleyecek. Bu hızı da bilmesi için kullanıcıya 7-segment'ler aracılığıyla motorlarımızın hızını göstereceğiz. Çoklu motor olması amacıyla uygulamamızda 3 motor ve bu 3 motorun hızını gösterecek 6 tane de 7-segment display kullanmaya karar verdik. Unutmadan; hızı kullanıcı belirleyecek diyorduk, onun için de 6 tane hız arttırma ve azaltma düğmesi kullanacağız.

Bu uygulamanın ileriki robotik çalışmalarda size faydasının dokunması dileğiyle ☺ iyi çalışmalar...

## Pulse Width Modulation (Darbe Genişliği Modülasyonu)

Başlangıç olarak motorların hızını nasıl kontrol edebileceğimizi incelemeliyiz. Kullandığımız mikrodenetleyicinin çıkışlarından logic olarak 1 veya 0, elektriksel karşılığı olarak da +5V veya 0V alabiliriz. Fakat bu uygulamamızda olduğu gibi, çeşitli uygulamalarda bu iki seviye arasında kalan herhangi bir değeri çıkış olarak almamız gerekebilir. Bu durumda başvuracağımız yöntemlerden birisi Pulse Width Modulation (PWM)'dir.

PWM'ı örneklemek gerekirse; düz bir sokakta kay kay ile kayan bir çocuğu düşünün. Ayağıyla yeri ittiği anı +5V, kayağın üzerinde gittiği anı da 0V olarak kabul edelim. Bu çocuğun ivmesini belirleyecek olan faktör birim zamanda yeri ne kadar ittiği olacaktır. Eğer yeri sürekli itiyorsa ivmesi maksimum (yani bizim sistemimize göre +5V) eğer yeri hiç itmiyorsa ivmesi 0 (sistemimize göre ise 0V) olacaktır. Fakat bu çocuk toplam zamanın yarısı kadar süre ile yeri itiyorsa *ortalama ivmesi* maksimum ivmesinin %50'si olacaktır. Sistemimize dönecek olursak; bu bizim için ortalama olarak 2.5V çıkışa denk gelmektedir. Görüldüğü gibi çocuğun (mikrodenetleyici'nin) yeri ittiği (+5V çıkış verdiği) sürenin toplam süreye oranı çocuğun ortalama ivmesini (mikrodenetleyicinin ortalama çıkışını) verecektir.



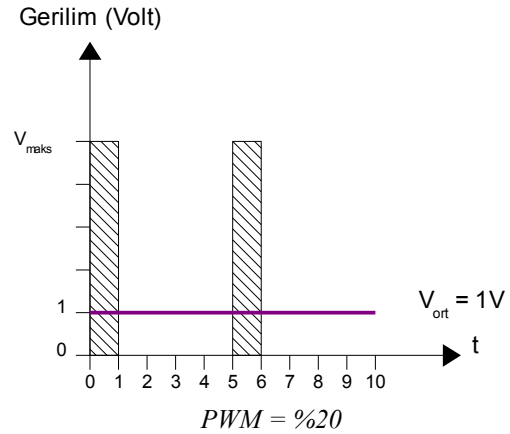
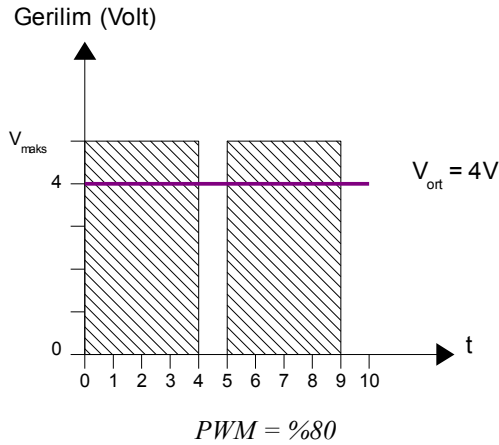
Şekilde de görüldüğü gibi toplam 10 birim zamanın 6 biriminde çıkış olarak +5V verilmiştir.

$$Ortalama \text{ Çıkış} = \frac{5V \times 6 \text{ Birim}}{10 \text{ Birim}} = 5V \times \%60 = 3V$$

Denklemden verildiği gibi maksimum çıkış değerinin %60'ını ortalama çıkış değeri olarak görüyoruz. %60'ı *PWM oranı* olarak da adlandırmamızda bir sakınca yoktur.

$$PWM \text{ Oranı} = \frac{\text{Maksimum Olma Süresi}}{\text{Toplam Süre}}$$

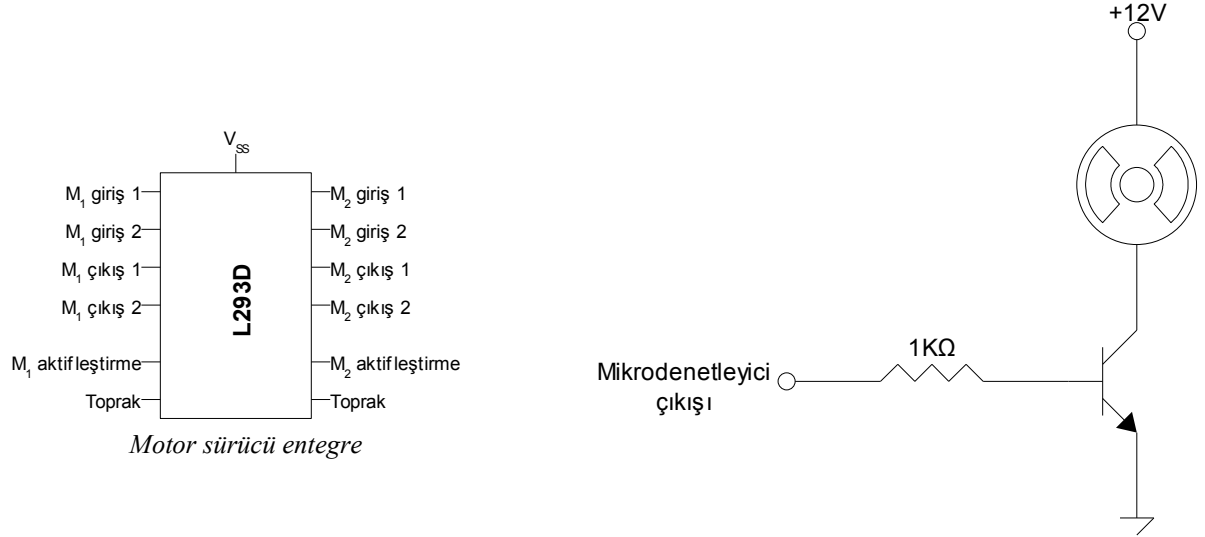
Uyguladığımız çıkış sinyalinin  $V_{maks}$  olma oranı arttıkça ortalama çıkış voltajımız da artacaktır.



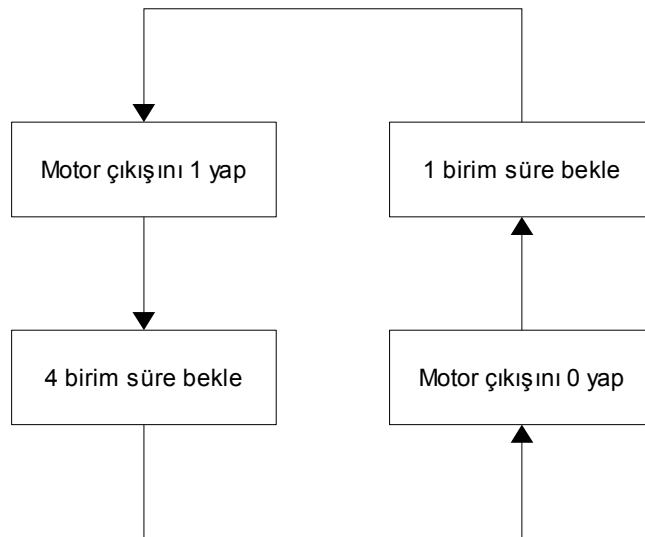
Çıkış sinyalinin periyodunu motorun tepki süresinden kısa seçtiğimiz vakit, çıkış sinyalinin ortalama değerini motor üzerinde durağan bir biçimde görebiliriz. Ancak seçtiğimiz periyodun çok uzun olması durumunda sinyaldeki değişimleri motor üzerinde doğrudan görebiliriz, bu da motorun kesik kesik hareket etmesine neden olacaktır.

Motoru doğrudan mikrodenetleyicinin çıkışına bağladığımız zaman mikrodenetleyicinin çıkışından motoru istediğimiz hızda çevirecek kadar akım alamayız. Çıkışı +5V olarak kabul eden mikrodenetleyici gerekli akımı sağlamaya çalışacak ancak başarılı olamayınca kullanılamaz hale

gelecektir. Aynı zamanda  $V_{maks}$  olarak +5V kullanmak yerine daha küçük yada daha büyük voltaj ile motoru sürmek isteyebiliriz. Bu yüzden mikrodenetleyici ile motor arasında bağlantı kuracak bir elemana ihtiyacımız var. Ara eleman olarak motor sürmek için hazırlanmış olan motor sürücü entegreleri kullanabileceğimiz gibi transistörlerin anahtarlama özelliğinden de faydalanabiliriz.

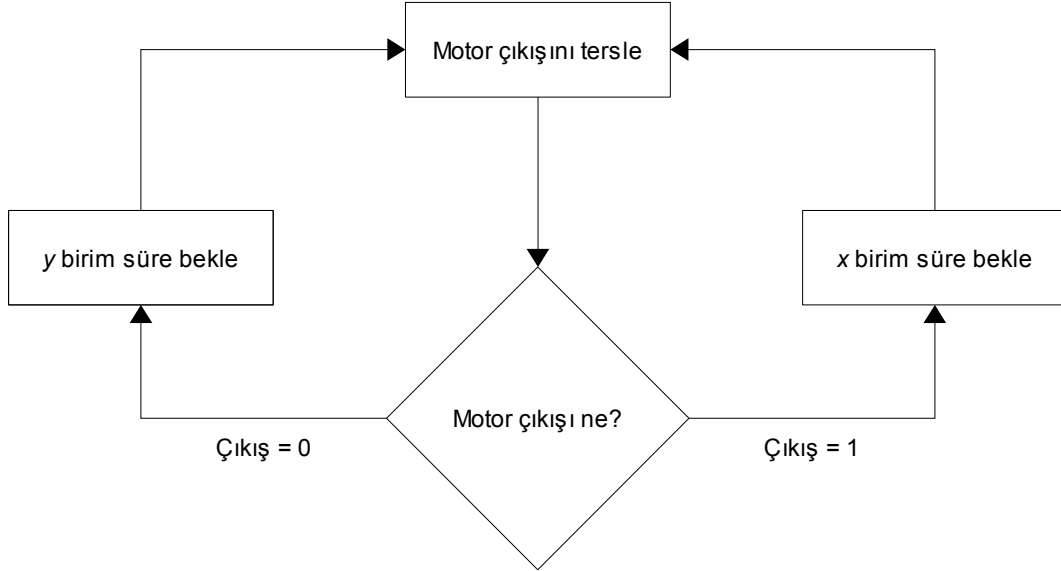


Temel olarak transistörden bahsetmek gerekirse; NPN tipi transistörün base'i ile emiter'i arasına 0,6V (ve fazlası) uygulanırken transistörün collector'ü ile emiter'i arasında akım geçişi (bağlantı) sağlanır. Eğer base ile emiter arası gerilim 0,6V'un altına inerse, collector ve emiter arasındaki akım geçişi kesilir. Bu sayede transistöre verdiğimiz sinyalin şekil olarak aynısını motor üzerinde görebiliriz. Ancak motor üzerinde görülebilecek  $V_{max}$  motorun diğer ucunun bağlı olduğu voltaj kaynağına göre değişir. Bu özelliğiyle çıkış gücü +5V olan mikrodenetleyici ile +12V'la çalışan bir motoru sürebiliriz. Şimdi bu çıkış sinyallerini, kullandığımız mikrodenetleyici ile nasıl verebileceğimize bakalım.



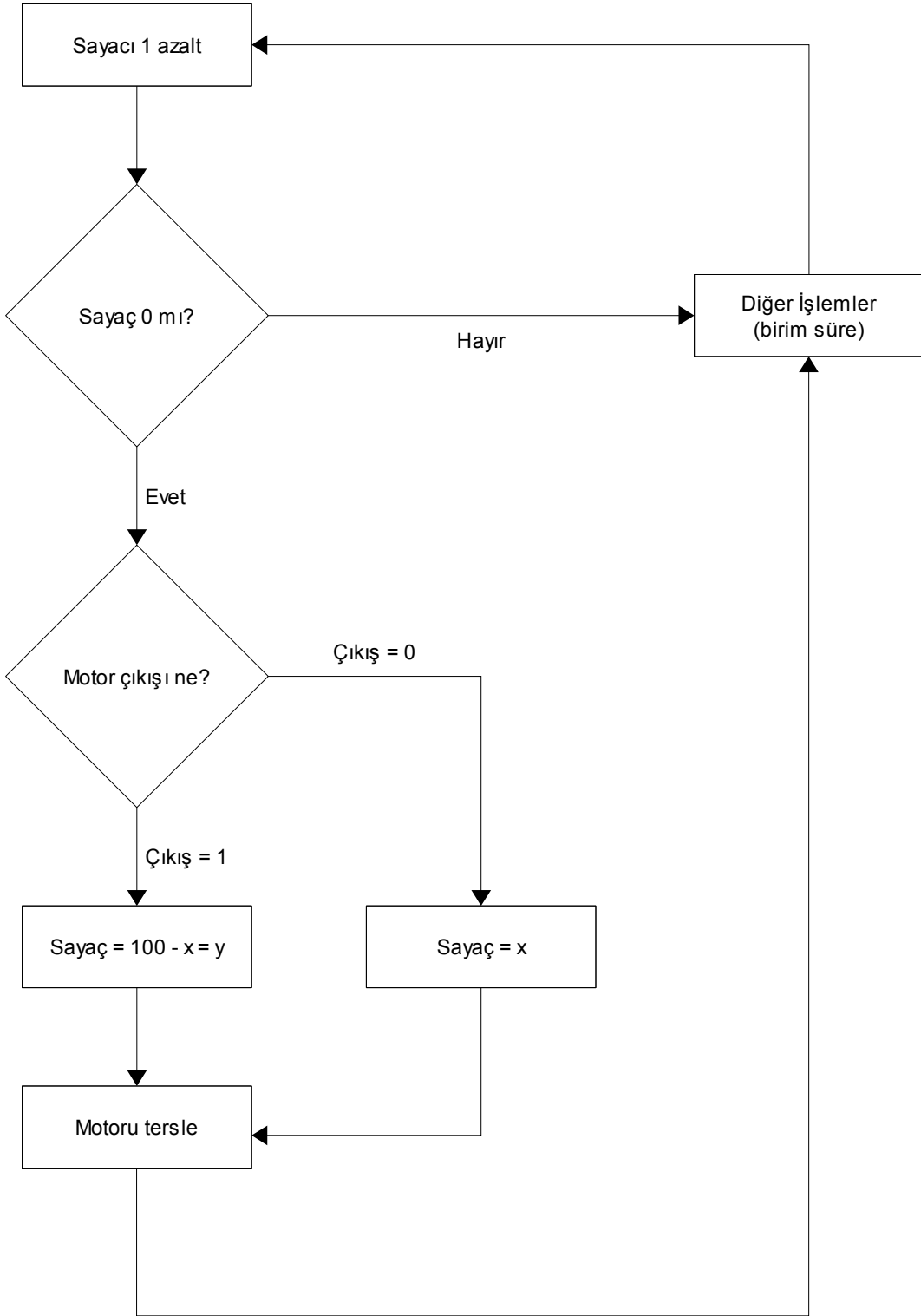
Yukarıda verilene benzer bir algoritma mikrodenetleyiciden almak istediğimiz sinyali verebilir (PWM oranı %80). Verilen algoritmada bekleme sırasında istediğiniz işlemleri yapmak mümkün. Beklemeler esnasında 7-segment display'lere motor hızlarını yazabiliriz.

Motor PWM kontrolünü sağladığımız bu algoritmayı biraz daha sadeleştirelim.



İlk yazdığımız algoritma 4 seviyeden oluşuyordu ama ikinci yazdığımız algoritma 3 seviyeden oluşuyor. Böylelikle programımızı fazla kodlardan kurtarmış olduk. Önceden de değindiğimiz gibi mikrodenetleyicinin yapacağı tek iş bu PWM'ı ayarlamak olmayacaktır. Bu yüzden algoritmamızı biraz daha genelleştirmemiz ve diğer işlemleri de hesaba katacak şekilde genişletmemiz gerekiyor.

Mikrodenetleyicinin diğer işlemlerde (2. ve 3. motorun PWM çıkışı, 7-segment display'lerde gösterim, buton kontrolü) sabit vakit harcadığını ve geçen toplam zamanın da birim zaman olduğunu kabul edelim. Bu durumda motor çıkışı 1 (+5V) iken 4 defa (x defa), motor çıkışı 0 olunca 1 defa (y defa) diğer işlemleri yapacak ve tekrar motor pwm çıkışı ayarlamasını yapacak.



Bu aşamada bir önceki alırtmada bulunan motoru tersle işleml ile motor çıkışını kontrol etme aşamasının yerini deęiřtirdik. Motor çıkışı ne diye sorduęumuzda eęer çıkış 1 ise sayaca attıęımız deęer motorun sıfır kalma süresi olmalıdır çünkü hemen bir sonraki işlemlde motoru tersleyeceęiz.

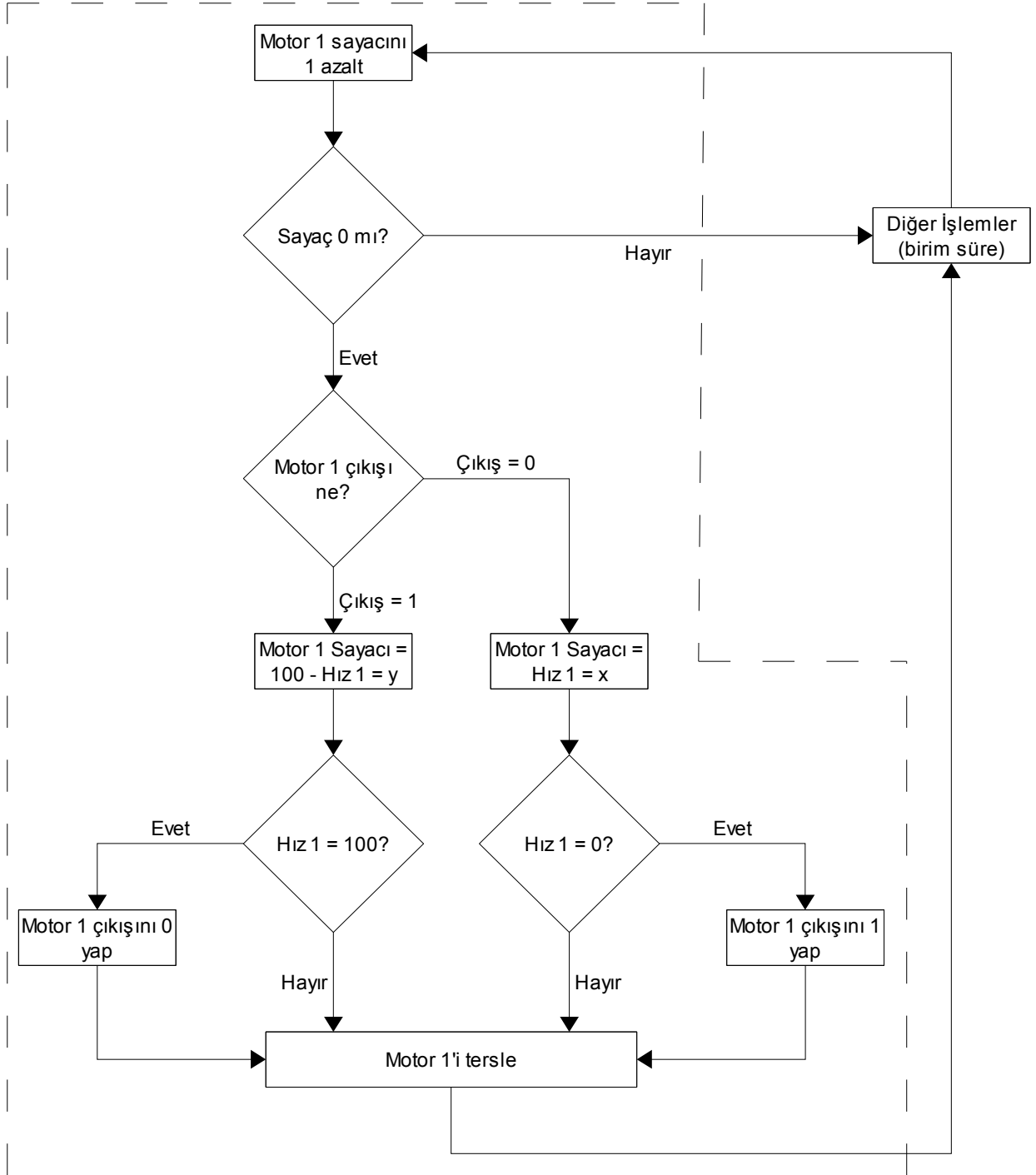
Eğer motor çıkışı 1 ise mikrodenetleyici *Sayaç* değişkeninin içine  $y$  değerini atayacak ve ardından motor çıkışını tersleyerek 0 yapacaktır. Mikrodenetleyici, diğer işlemleri gerçekleştirdikten sonra *Sayaç* değişkenini bir azaltacak, değişken 0 olana kadar diğer işlemleri tekrarlayacaktır. Yani mikrodenetleyici *Sayaç* değişkeni 0 olana kadar toplamda  $y$  kere diğer işlemleri yapacak. Diğer işlemlerde harcadığımız süreyi birim süre olarak kabul etmiştik. Bu durumda motor çıkışımız terslenip 0 olduktan sonra  $y$  birim süre beklemiş olacağız.

Algoritmamızı bir aşama daha ilerletmiş olduk. Artık “diğer işlemler” diye belirttiğimiz kısımda istediğimiz işlemleri yapabileceğiz.

Hız ( $V_{ort}$ ) ölçeğimizi 100 olarak alacak olursak  $Y=100-X$  olacaktır. Çıkışın 1 olma oranı %80 ise 0 olma oranı %20'dir. Başka bir deyişle, 80 birim zaman boyunca çıkış  $V_{maks}$  ise 20 birim zaman

boyunca 0'dır.  $X$  bize  $\frac{V_{ort}}{V_{maks}}$  oranını, yani hız oranımızı verdiğine göre  $X$ 'i bundan sonra *hız değişkeni* olarak tanımlayabiliriz.  $X$ 'e verdiğimiz değer arttıkça PWM'deki 1 süresi artacak ve bu artış da beraberinde motora uygulanan ortalama voltajı yükselterek motorun hızını arttıracaktır.

Artık uygulamamızda kullanacağımız PWM algoritmasının sonuna geldik. En son yazdığımız algoritma 2 PWM değeri hariç tüm PWM'larda sorunsuz çalışacaktır. PWM oranı 0 veya 100 olduğunda algoritmada sorun olmaması için bu iki durum için özel ekleme yapmalıyız.



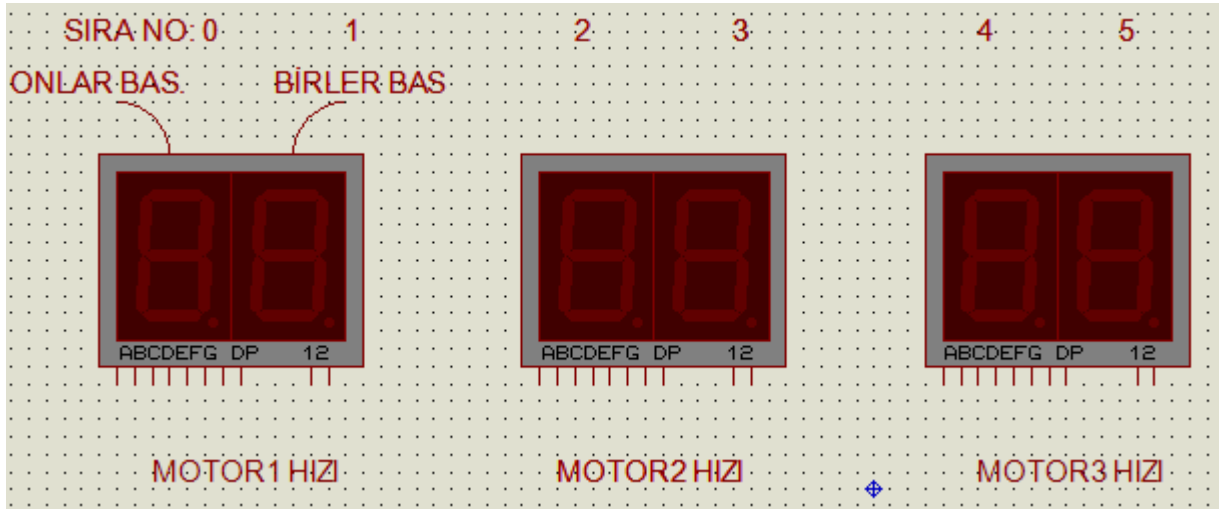
Böylelikle uygulamamızda PWM çıkışı üretecek algoritmamızı yazmış olduk. Kesikli çizgilerle belirtilen kısmı motorlara bağlı olarak değişen değişkenleri değiştirerek 2. ve 3. motor için tekrarlayabiliriz. Makalenin buradan sonraki bölümlerinde kesikli çizgilerle belirtilen kısmı 3 motoru da kontrol ettiğimizi kabul ederek *motor kontrol* olarak adlandıracağız.

## 7-Segment Display'lerde Gösterim

7-segment display'lerin yapısı ve de çoklu 7-segment'lerin kullanımı hakkındaki bilgilere [http://robot.ee.hacettepe.edu.tr/v5.0/dosyalar/makaleler/9\\_geri\\_sayim\\_cihaz.pdf](http://robot.ee.hacettepe.edu.tr/v5.0/dosyalar/makaleler/9_geri_sayim_cihaz.pdf) adresindeki makalemizden ulaşabilirsiniz. Yüzeysel olarak bahsetmemiz gerekirse, 7-segment display içerisinde 8 adet led bulunur. Bu led'lerin 7 tanesi ekranda sayının gösteriminde kullanılırken 1 tanesi ondalık noktası olarak kullanılır. Led'lerin iletme geçerek ışık verebilmesi için led'in anoduna +5V, katotuna 0V vermeniz gerekmektedir. 7-segment'lerde bulunan led'lerin bu iki bacağından birisi ortak bağlanmıştır. Örneğin ortak katot 7-segment display içerisindeki led'lerin katot bacakları ortaktır. Bu ortak bacağa 0V verdiğiniz zaman hangi led'e +5V verirseniz 7-segment'in o kısmı yanacaktır.

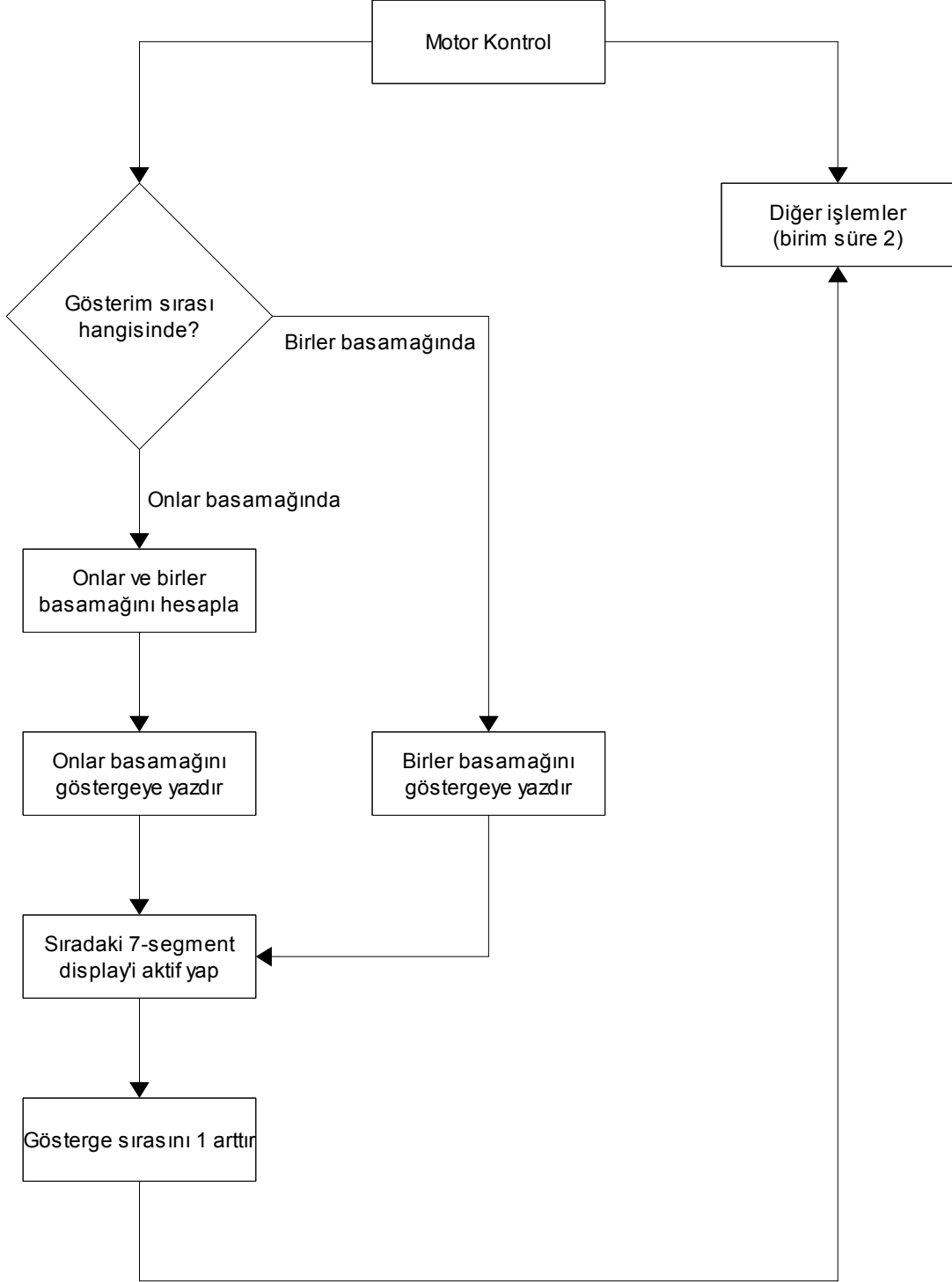
Çoklu 7-segment kullanırken çalışan 7-segment'i ortak bacağına verdiğimiz voltaj ile değiştirebiliriz. Örnek olarak 3 tane ortak katotlu 7-segment düşünün. Sadece ortak bacağına 0V verdiğimiz 7-segment'ler çalışacaktır. Diğer 7-segmentler ise sönük kalacaklardır. Yani programda yapacağımız şey, 7-segment'te hangi sayı gösterilecekse o gösterime uygun bir şekilde 7-segment'in giriş uçlarına +5V yada 0V vermek ve de çalışacak olan 7-segment'in ortak bacağına 0V, diğer 7-segmentlerin ortak bacağına +5V vermek olacaktır. Bu işlem sıra ile diğer 7-segment'lere de uygulanmalıdır.

Uygulamamızda 3 motorun hızlarını göstermek istiyoruz. Hız ölçeğini önceden 100 kabul ettiğimiz için her motorun hızını gösterebilmek için 2'ser adet 7-segment display'e ihtiyacımız olacaktır. Toplamda kullanacağımız 7-segment sayısı böylece 6 oldu.

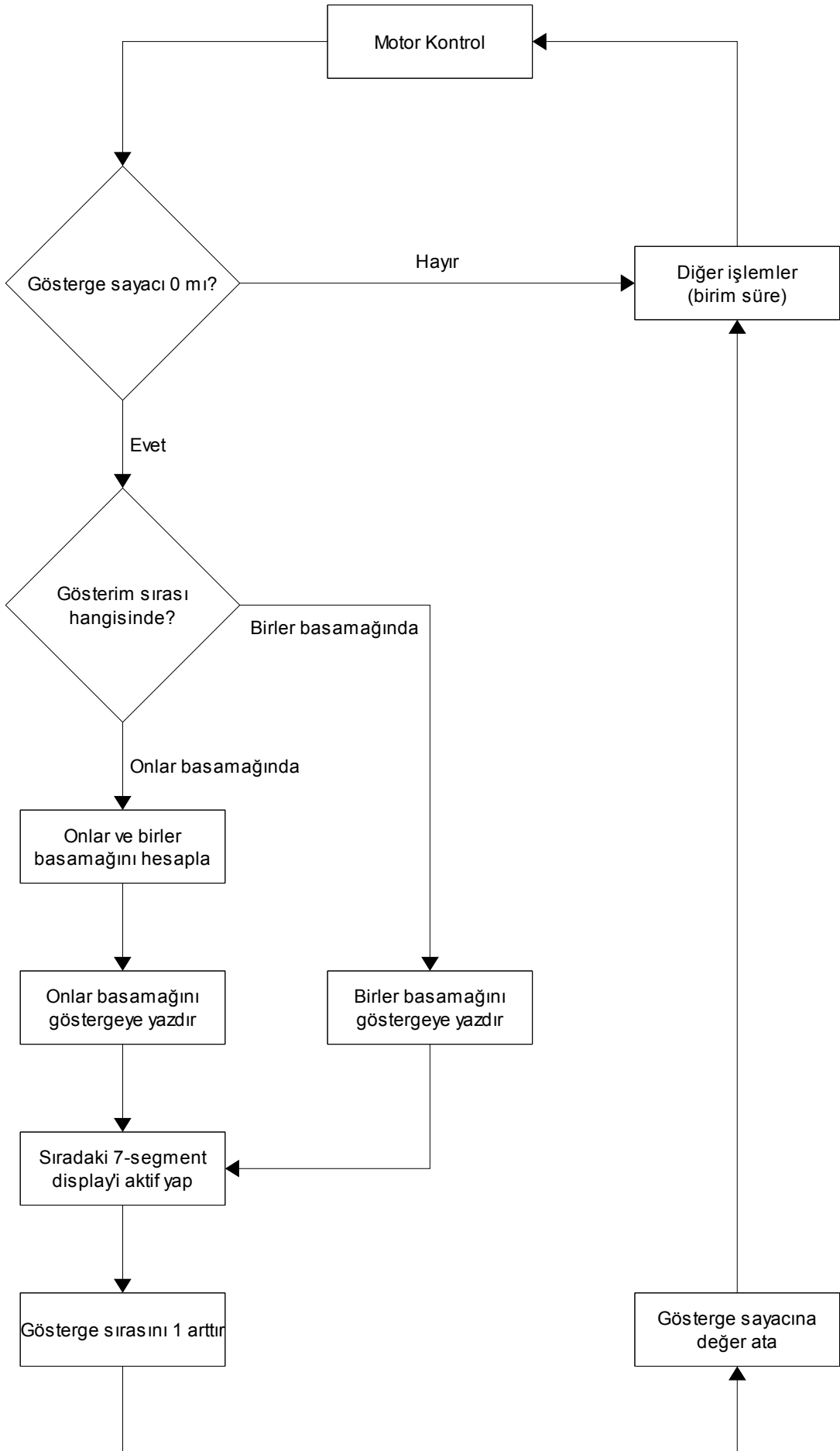


Bu 7-segment'lere 0-5 arası sıra ile numara verelim. Sıra numarası hangi 7-segmenti çalıştıracığımız ve hangi rakamı yazdıracağımızı belirtecek. Göstergeleri çalıştırma sırasının 0'dan 5'e doğru olduğunu kabul edersek motor hızının önce onlar basamağı sonra da birler basamağı ekrana yazdırılacak demektir. Hazırlayacağımız algoritmada hızların onlar basamağını

ve birler basamağını hesaplayacak işlemler de yapmalıyız çünkü hız değişkenimizde ölçek olarak 100 kullanmıştık. Eğer 1. motorun hızı 67 gibi bir değerse, hazırlayacağımız algoritma 0 nolu 7-segment'te 6 rakamını, 1 nolu 7-segment'te 7 değerini göstermelidir. Önce onlar basamağını ekranda göstereceğimiz için bu sayı değerlerini her motorun onlar basamağını gösterirken hesaplamamız bizim için daha kolay, programımız için daha kullanışlı olacaktır.

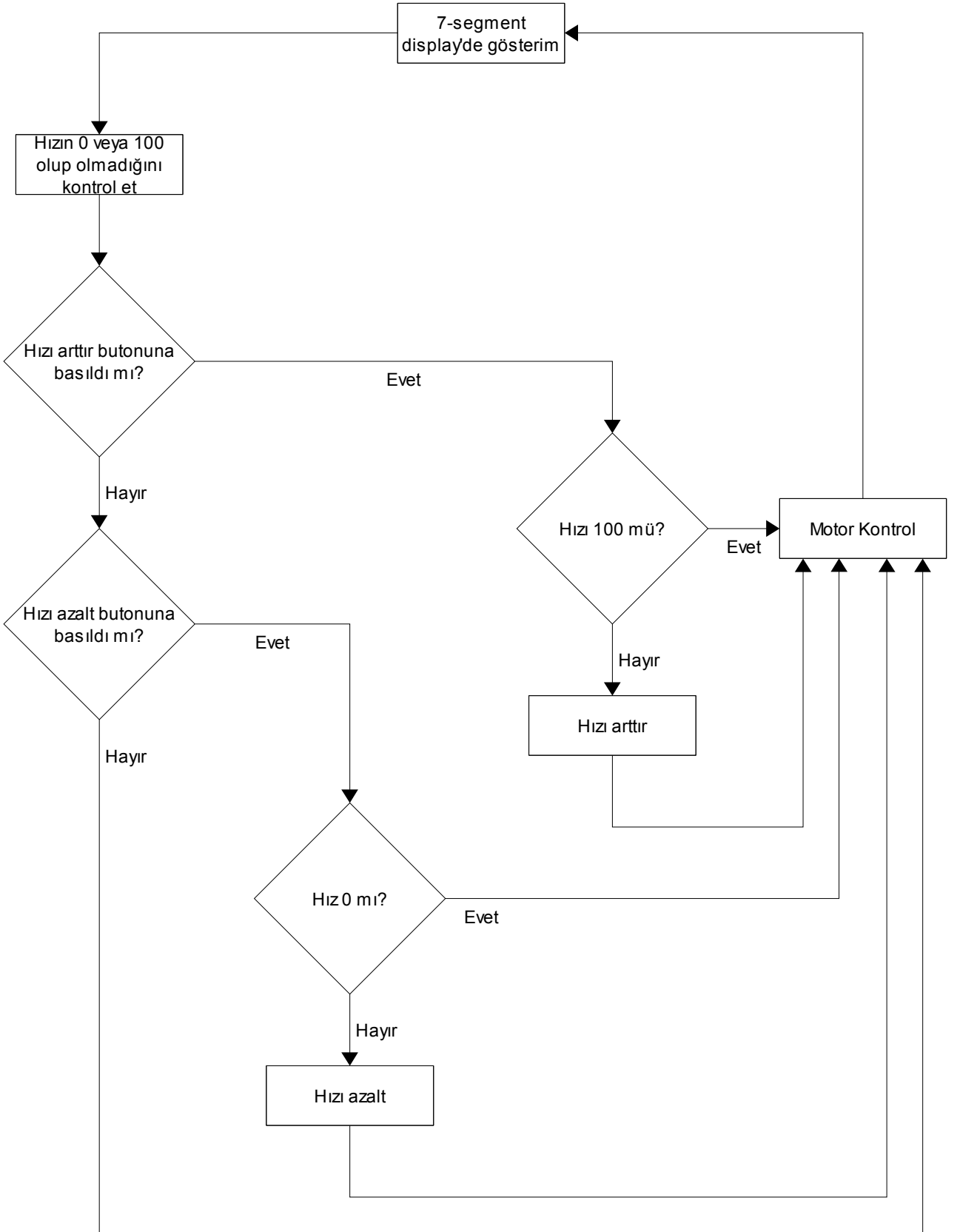


Bu algoritma işimizi görecektir ancak çok hızlı bir şekilde gösterge sırasını değiştirirse sayılar 7-segment'lerde sönük görünecektir. Aynı şekilde gösterge sırası çok yavaş değişirse sayılar 7-segment'lerde sıra ile görünecektir. Bu iki istemediğimiz koşulu ortadan kaldırmamız gerekiyor. İlk koşul için motorda kullandığımız yöntemi kullanabiliriz. Bir sayaç yardımı ile belli defa motor kontrol + diğer işlemleri yaptıktan sonra gösterge değişimini gerçekleştiririz. İkinci koşul içinse diğer işlemler + motor kontrol kodlarının çok uzun süren işlemler olmaması gerekmektedir. 4MHz ile çalışan mikrodenetleyicimizin 1 işlemi 1  $\mu$ s'de yaptığını düşünecek olursak yaptığımız uygulama zaten çok kısa sürecektir. O halde yeni algoritmamızı yazalım.



## Buton Kontrol

Gösterge algoritmamızı da yazdığımızı göre geriye sadece diğer işlemler diye tabir ettiğimiz bölümde hız butonlarını kontrol etmek kaldı. Hız butonlarını kontrol edeceğiz, eğer bir motorun hız arttır butonuna basılmışsa ve o motorun hızı zaten 100 değilse motorun hız değişkenini arttıracacağız. Eğer hız azalt butonuna basılmışsa bu durumda da hızın 0 olup olmadığını kontrol ettikten sonra hızını azaltacağız. Bu işlem bu kadar basit.



Önceki algoritmalarda yaptığımız gibi buton kontrol aşamasına da bir sayaç yerleştirmeliyiz. Mikrodenetleyicimiz çok hızlı işlem yaptığı için kullanıcı elini butondan çekmeden hız

değişkenimiz 100'e ulaşabilir. Bu yüzden bir sayaç yardımıyla belirli sürede bir butonları kontrol edebiliriz. Bu sayacı da ekledikten sonra programımızın algoritmasını bitirdik demektir.

Algoritmamızla ilgili değinmemiz gereken çok önemli bir nokta daha var. Dikkat ederseniz *diğer işlemler* olarak tabir ettiğimiz işlemleri yani motor kontrolü sırasında *buton kontrol + 7-segment gösterim*, 7-segment gösterim sırasında *motor kontrolü + buton kontrol*, buton kontrol sırasında ise *motor kontrol + 7-segment gösterim* işlemlerini hep sabit süreli olarak kabul ettik ve programımızı ona göre hazırladık. 7-segment gösterimde pek önemli olmasa da PWM çıkışı ürettiğimiz motor kontrol işlemlerinde zamanlama oldukça önemlidir. Bu yüzden hazırladığımız program ana döngüde 1 tur attığında her koşulda aynı zamanı harcamalıdır. Bunun için de kullandığımız sayaçlar sıfırlanıp *motor*, *gösterim*, *buton kontrol* işlemlerine girse de girmese de aynı zamanda döngüyü tamamlayabilmesi için gerekli yerlere gereken gecikme işlemlerini koymalıyız. Bu gecikmelerin uzunluğunu programı yazdığımız zaman kontrol edebiliriz.

## Assembly Programı

```

list          p=16f877A
#include      <p16f877A.inc>

        __CONFIG __CP_OFF & __WDT_OFF & __BODEN_OFF & __PWRTE_ON & __XT_OSC &
__WRT_OFF & __LVP_OFF & __CPD_OFF

; PORTA motorlar, PORTB 7-segment, PORTC butonlar, PORTD 7-segment degisimi

w_temp          EQU    0x71
status_temp     EQU    0x72
pclath_temp     EQU    0x73
BUTONSAYACI    EQU    0X20
KONTROLTEMP     EQU    0X21
BUTONTEMP      EQU    0X22
HIZ1 EQU       0X23
HIZ2 EQU       0X24
HIZ3 EQU       0X25
M1SURESI      EQU    0X26
M2SURESI      EQU    0X27
M3SURESI      EQU    0X28
GOSTERGE_SIRASI EQU    0X29
BIRLER_BAS    EQU    0X2A
ONLAR_BAS     EQU    0X2B
GOSTERGE_SAYAC EQU    0X2C
TEMP EQU      0X2D

#define        M1 PORTA, 0
#define        M2 PORTA, 1
#define        M3 PORTA, 2
#define        HIZART1      PORTC, 0

```

```
#DEFINE HIZAZ1 PORTC, 1
#DEFINE HIZART2 PORTC, 2
#DEFINE HIZAZ2 PORTC, 3
#DEFINE HIZART3 PORTC, 4
#DEFINE HIZAZ3 PORTC, 5
```

```
ORG 0x000
```

```
NOP
goto main
```

```
ORG 0x004
```

```
movwf w_temp
movf STATUS,w
movwf status_temp
movf PCLATH,w
movwf pclath_temp
```

```
movf pclath_temp, w
movwf PCLATH
movf status_temp, w
movwf STATUS
swapf w_temp, f
swapf w_temp, w
retfie
```

LUKAP

```
ADDWF PCL, F
RETLW B'00111111'
RETLW B'00000110'
RETLW B'01011011'
RETLW B'01001111'
RETLW B'01100110'
RETLW B'01101101'
RETLW B'01111101'
RETLW B'00000111'
RETLW B'01111111'
RETLW B'01101111'
RETLW B'10000110' ; 10 ise 1. şekilde görülecek.
```

GOSTERGE\_GECIKME\_TAMAMLAMA

```
MOVWF TEMP
MOVLW D'245'
ADDWF TEMP, F
GOTO $+1
NOP
```

```

INCFSZ     TEMP
GOTO  $-3
RETURN

```

## ONMIKROGECIKME

```

MOVLW D'1'
MOVWF 0X74
DECFSZ     0X74
GOTO  $-1
GOTO  $+1
RETURN

```

## BIRLER\_GECIKME

```

MOVLW D'29'
MOVWF TEMP
DECFSZ     TEMP
GOTO  $-1
GOTO  BIRLER_GOSTER

```

## GOSTERGEYE\_YAZDIR

```

MOVWF TEMP ; Bu altprograma girdiğimizde her seferinde W'da bir
; hız değeri bulunacaktır. Bunu TEMP değişkenine
; atıyoruz.

CLRF ONLAR_BAS ; Gerekli değişkenler sıfırlanıyor.
CLRF BIRLER_BAS
MOVLW D'10' ; W'ya 10 değerini verdik.
SUBWF TEMP, F ; TEMP'ten 10 değerini çıkarıp TEMP'in içine
; yazıyoruz. PIC'in komutları arasında bölme
; işlemi olmadığı için çıkarma işlemini
; kullanarak bölme yapıyoruz diyebiliriz. Kaç
; çıkarma işleminde sonuç pozitifse bu bize hız
; değişkenlerimizin birisinin onlar basamağını
; verecek.

BTFSS STATUS, C ; Sonuç negatif mi?
GOTO $+3 ; Evet, 3 satır aşağıdan devam et.
INCF ONLAR_BAS ; Hayır, ONLAR_BAS değişkenini bir artır.
GOTO $-4

ADDWF TEMP, W ; Son işlemde a sayısından 10 çıkarınca sonuç negatif
; olmuştu. Demek oluyor ki a sayısı 10'dan küçük yani
; bizim birler basamağımız. TEMP'te bulunan negatif
; sayıya W'da bulunan 10'u eklersek birler basamağını
; buluruz.

MOVWF BIRLER_BAS ; Birler basamağını da BIRLER_BAS değişkenine yazdık.
; Bu noktaya kadar gösterge hesaplamalarını yaptık.
; Görüldüğü gibi herhangi bir gösterme işlemi
; yapılmadı. Sıra ona geldi.

MOVFW ONLAR_BAS ; W'ya onlar basamağını attık ve 2 komut satırı aşağı

```

## HUNRobotX - Makaleler - Hız Ayarlı Çoklu DC Motor Kontrolü

```
                ; indik.
CALL  GOSTERGE_GECIKME_TAMAMLAMA
MOVFW ONLAR_BAS
GOTO  $+2

BIRLER_GOSTER
MOVFW BIRLER_BAS ; Eğer onlar basamağını W'ya atmışsak program bu satırı
                ; okumayacak.
CALL  LUKAP      ; W'daki değer LUKAP tablosuna gönderildi.
MOVWF PORTB     ; LUKAP tablosundaki değeri PORTB'ye yazdırdık.
GOTO  GOSTERGEYE_DEVAM

KONTROL
BCF  KONTROLTEMP, 0
BCF  KONTROLTEMP, 1 ; Değerler sıfırlandı.
BTFSC STATUS, Z    ; Kontrole gönderilen hız değeri sıfır mı?
BSF  KONTROLTEMP, 1 ; Evet, KONTROLTEMP'in 1. bit'ini 1 yap.
SUBLW D'100'
BTFSC STATUS, Z    ; Hız'dan 100 çıkar ve sıfır mı diye kontrol et.
BSF  KONTROLTEMP, 0 ; Evet, KONTROLTEMP'in 0. bitini 1 yap. Yani hız
                ; 100'müş

RETURN

HIZBUTONKONTROL
MOVFW HIZ1
CALL  KONTROL    ; HIZ1'i kontrole gönderiyoruz (kontrole bakınız).
BTFSC HIZART1   ; HIZ1'i arttır butonu basılı değilse 1 satır atla,
                ; basılıysa aşağıdaki komutu işle.
BTFSC KONTROLTEMP, 0 ; HIZ1 100 mü? 100 ise aşağıdaki komutu işle,
                ; değilse bir satır atla.

GOTO  $+2
INCF  HIZ1      ; HIZ1 100 ise bu komut işlenmeyecek.

BTFSC HIZAZ1    ; HIZ1'i azalt butonu basılı mı?
BTFSC KONTROLTEMP,1 ; HIZ1 sıfır mı? Sıfırsa bir aşağıdaki satırı
                ; işle, değilse bir satır atla.

GOTO  $+2
DECF  HIZ1      ; HIZ1 sıfırsa bu komut işlenmiyor.

MOVFW HIZ2
CALL  KONTROL
BTFSC HIZART2
BTFSC KONTROLTEMP, 0
GOTO  $+2
INCF  HIZ2

BTFSC HIZAZ2
```

```
BTFSC KONTROLTEMP, 1
GOTO $+2
DECF HIZ2

MOVFW HIZ3
CALL KONTROL
BTFSC HIZART3
BTFSC KONTROLTEMP, 0
GOTO $+2
INCF HIZ3

BTFSC HIZAZ3
BTFSC KONTROLTEMP, 1
GOTO $+2
DECF HIZ3
NOP
RETURN
```

main

```
BCF STATUS, RP0
BCF STATUS, RP1
CLRF PORTA
CLRF PORTB
CLRF PORTC ; Portlar sıfırlandı.
BSF STATUS, RP0
CLRF TRISA
CLRF TRISB
CLRF TRISD
MOVLW D'255'
MOVWF TRISC ; Gerekli yönlendirmeler yapıldı.
MOVLW B'0111'
MOVWF ADCON1 ; PORTA varsayılan olarak analog giriş olarak
; ayarlı. PORTA'yı çıkış yaptıktan sonra emin olmak
; için A/D modülünü kapattık.

BCF STATUS, RP0
BCF ADCON0, 0

MOVLW D'32'
MOVWF PORTD

MOVLW D'6'
MOVWF TEMP
MOVLW H'23'
MOVWF FSR
MOVLW D'50'
MOVWF INDF
```

## HUNRobotX - Makaleler - Hız Ayarlı Çoklu DC Motor Kontrolü

```
INCF FSR
DECFSZ TEMP
GOTO $-4 ; Peşpeşe birkaç değişkene aynı değeri atamak gerekiyordu
; FSR'yi de programda kullanmış olalım dedim.
MOVLW D'250'
MOVWF GOSTERGE_SAYAC
MOVLW D'250'
MOVWF BUTONSAYACI
CLRF GOSTERGE_SIRASI
CLRF BIRLER_BAS
CLRF ONLAR_BAS
```

### ANADONGU

```
; Bu anadöngü her tamamlandığında her koşulda eşit süre geçmektedir. Bu süre
; 228us (4MHz kristal için)'dir. Biraz sonra göreceğiniz BUTONSAYACI
; değişkeni kaç 228 mikrosaniyede bir butonları kontrol edeceğimizi
; belirlemeye yarar. Daha sonra göreceğiniz GOSTERGE_SAYAC değişkeni de
; benzer bir işlem gerçekleştirmektedir.
```

```
DECFSZ BUTONSAYACI
GOTO $+5
CALL HIZBUTONKONTROL ; (65 mikrosaniye) bu rutinde butonlar kontrol
; ediliyor. Açıklamalar için alt rutine bakınız.
MOVLW D'250'
MOVWF BUTONSAYACI ; Sayaca tekrar değer verdik.
GOTO $+7 ; Ana döngüye devam ediyoruz.
MOVLW D'5' ; Bu satıra gelmesi demek butonları kontrol etmemiş
; olmamız demek. Ana döngünün her koşulda eşit süreyi
; vermesi için buton kontrol ederek geçirdiğimiz süreyi
; butonları kontrol etmediğimiz zaman da harcamalıyız.
; bu yüzden bu noktaya gecikme koyuyoruz.
MOVWF BUTONTEMP
CALL ONMIKROGECIKME
DECFSZ BUTONTEMP
GOTO $-2
GOTO $+1
```

```
; MOTOR RUTİNLERİ
; Bir motorun PWM'ını belirleyen motorun açık kalma ve kapalı kalma
; süreleridir. Açık kalma süresi/tüm süre bize motorun PWM'ını verecektir.
; Ana döngünün bu noktaya sürekli eşit sürede geleceğini düşünürsek ana
; döngünün attığı kaç turda kapalı, kaç turda açık kaldığı aynı şekilde bize
; PWM'ı verecektir. M1SURESI değişkeni ana döngü bu noktaya her geldiğinde
```

## HUNRobotX - Makaleler - Hız Ayarlı Çoklu DC Motor Kontrolü

; bir azalıyor ve kaç tur sonra girmesini istemişsek o noktada sıfırlanıp  
; motoru 1 yada 0 yapıyor. Motoru 1 veya 0 yaptıktan sonra motorun açık  
; ya da kapalı kalma süresine göre tekrar M1SURESI degiskenine sayı atıyor.  
; Ana döngü gerekli sayıda tur attıktan sonra yukarıdaki işlem tekrarlanıyor.  
; Programın bu kısmının da her koşulda eşit sürede işlenmesi gerekmektedir.  
; Bunun için gerekli yerlere gecikme yerleştirdik.

### MOTOR1ACKAPA

```
DECFSZ      M1SURESI
GOTO  MOTOR1ACKAPASONU ; Eğer motorda değişiklik yapılmayacaksa burada
                        ; harcayacağı zamanı geçirmek üzere gecikmeye
                        ; gidecek.
MOVFW HIZ1  ; Hız oranını W'ya attık.
BTFSZ M1    ; Motorun durumunu kontrol ediyoruz. Eğer motor açıksa
            ; M1ACIKTA'ya kapalıysa M1KAPALI satırlarına gidecek.
GOTO  M1ACIKTA
```

### M1KAPALI

```
BTFSZ STATUS, Z ; W'daki değeri yani hız oranınının sıfır olup olmadığını
                ; kontrol ediyoruz. Eğer sıfırsa motoru açıyoruz.
                ; Programın ilerleyen satırlarında değilini
                ; alma (tersleme) işlemi yaptığımız için motoru
                ; açıyoruz. 2 mikrosaniye sonra tekrar kapayacağız.
BSF  M1        ; Motorun hız oranı 0'sa değilini alma işleminde motorun
                ; sıfır kalması için bu noktada motoru açtık.
GOTO  SON1     ; W'daki değerimiz HIZ1 iken SON1'e gidiyoruz.
```

### M1ACIKTA

```
SUBLW D'100'    ; Motorun hız oranınının 100 olup olmadığını kontrol
                ; ediyoruz. Hız oranı 100 ise yukarıdaki açıklamaya
                ; benzer bir şekilde değilini alma işlemi öncesi
                ; motoru kapıyoruz.
BTFSZ STATUS, Z
BCF  M1        ; W'daki değerimiz 100-HIZ1 iken SON1'e gidiyoruz
```

### SON1

```
MOVWF M1SURESI ; W'daki değerimiz ya HIZ1'e yada 100-HIZ1'e eşittir.
                ; Eğer motorkapalı kısımdan geliyorsak W'da HIZ1 (yani
                ; açık olma oranı); motor açıktan geliyorsak 100-HIZ1
                ; (kapalı olma oranı) bulunuyor. Bu değerleri
                ; M1SURESI'ne atayarak motorun ana döngünün kaç turunda
                ; 1 yada 0 olacağını ayarlıyoruz.
MOVLW B'1'
XORWF PORTA    ; Motor 1'in bulunduğu pin terslendi.
GOTO  MOTOR2ACKAPA
```

### MOTOR1ACKAPASONU

```
CALL  ONMIKROGECIKME ; Eğer motorda değişiklik yapılmayacaksa yukarıda
                        ; harcayacağı zamanı burada harcayacak.
NOP
```

```
MOTOR2ACKAPA
    DECFSZ      M2SURESI
    GOTO  MOTOR2ACKAPASONU
    MOVFW  HIZ2
    BTFSC  M2
    GOTO  M2ACIKTA
```

```
M2KAPALI
    BTFSC  STATUS, Z
    BSF    M2
    GOTO  SON2
```

```
M2ACIKTA
    SUBLW  D'100'
    BTFSC  STATUS, Z
    BCF    M2
```

```
SON2
    MOVWF  M2SURESI
    MOVLW  B'10'
    XORWF  PORTA
    GOTO  MOTOR3ACKAPA
```

```
MOTOR2ACKAPASONU
    CALL  ONMIKROGECIKME
    NOP
```

```
MOTOR3ACKAPA
    DECFSZ      M3SURESI
    GOTO  MOTOR3ACKAPASONU
    MOVFW  HIZ3
    BTFSC  M3
    GOTO  M3ACIKTA
```

```
M3KAPALI
    BTFSC  STATUS, Z
    BSF    M3
    GOTO  SON3
```

```
M3ACIKTA
    SUBLW  D'100'
    BTFSC  STATUS, Z
    BCF    M3
```

```
SON3
    MOVWF  M3SURESI
    MOVLW  B'100'
    XORWF  PORTA
    GOTO  GOSTERGE
```

```
MOTOR3ACKAPASONU
    CALL  ONMIKROGECIKME
    NOP
```

; Tek tek açtığımız için göstergelere numara verdik.



## HUNRobotX - Makaleler - Hız Ayarlı Çoklu DC Motor Kontrolü

```
GOTO  GOSTERGEYE_YAZDIR      ; Program yukarılarda bulunan
                                   ; GOSTERGEYE_YAZDIR'dan devam etmektedir.
GOSTERGEYE_DEVAM
  RLF  PORTD
  MOVFW PORTD
  BTFSC PORTD,6
  MOVLW D'1'
  MOVWF PORTD

  MOVLW D'255'
  MOVWF GOSTERGE_SAYAC      ; Ana döngünün kaç turda gireceğini belirleyen
                                   ; register (önceden açıklamıştık).

  INCF  GOSTERGE_SIRASI

  GOTO  ANADONGU

GOSTERGE_GECIKMESI
  NOP      ; Program göstergeye girmemişse vakit harcamak için gecikme
                                   ; programı.
  MOVLW D'36'
  MOVWF TEMP
  DECFSZ   TEMP
  GOTO  $-1
  GOTO  ANADONGU
  END
```