MİKRODENETLEYİCİ KULLANILARAK FIRÇASIZ DC MOTORUN SENSÖRSÜZ KONTROLÜ

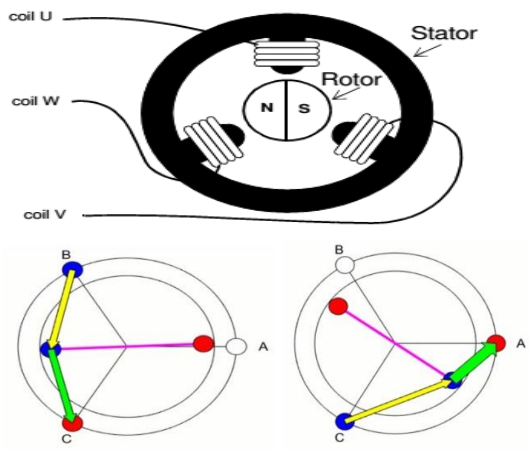
EREN ACAREL
İSTANBUL, 26/02/2021

I. Özet:

Sürekli mıknatıslı fırçasız DC motorlar, yüksek momentleri, güç ağırlık oranı ve verimlerinden dolayı her türlü sektörde sıkça tercih edilmektedir. Fırçasız DC motorlar yapısı gereği elektronik olarak kontrol edilir ve kontrolünde doğru akım komütasyonu için rotor pozisyon bilgisini gerektirirler. Bununla birlikte, pozisyon sensörlerinin maliyeti ve güvenilirlik problemleri çoğu kişiyi sensörsüz fırçasız DC motor sürücüleri üzerinde çalışmaya sevk etmiştir. Ben de bu projede ilk olarak fırçasız DC motorların çalışma prensibini anlamaya çalıştım. Ardından da edindiğim bilgilerin ışığında fırçasız DC motorları sürebilmeye uygun olacak bir sürücü devresi(ESC) tasarlamaya çalıştım.

II. Fırçasız DC Motorların Mekeanik Yapısı

Genel olarak fırçasız DC Motorlarda **stator** ve **rotor** olarak adlandırılan iki parça bulunur. Stator, DC fırçasız motorun içinde sabit olarak bulunur. Üzerinde düzenli sarılı bobinler bulunur. Bobinler rotorun dönmesi için manyetik alan sağlar. Rotor tarafından elde edilen manyetik alan ile stator tarafından elde edilen manyetik alan aynı frekansta döner. Statorun üzerinde bulunan iki zıt bobini tek bir bobin gibi algılayarak zıt kutup üretilir ve bu iki zıt bağlantı sayesinde rotoru bir kutup çekecek, diğeri itecektir.



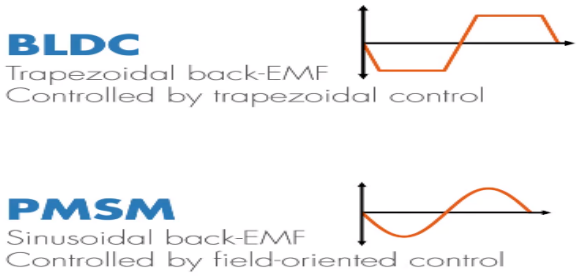
Rotor ise fırçasız DC motorlarda hareketli kısımdır. Rotor mili denen kısımda mıknatıslı bir yapı bulunmaktadır.

Ancak rotorun dışta ve statorun içte olduğu tasarımlar da mevcuttur. Bu tip farklılıkları motorun soğuması, eylemsizliğin düşüklüğü, mıknatısın dönme hızına göre kırılıp kırılmaması gibi durumlara bağlamaktayız. Bu kısma fazla değinmeyeceğiz.

III. Fırçasız Doğru Akım Motor (BLDC) Kontrolü ve Sürücü Devresi Tasarımı:

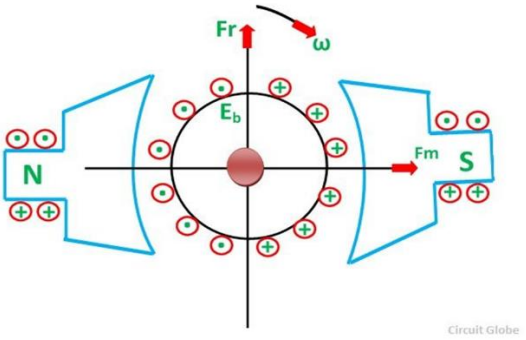
İsminden de anlaşılacağı üzere, tasarımında fırça yapısı bulunmamaktadır. Dönüş hareketi yapan parçasında (rotor) sabit mıknatıslar bulunmaktadır. Karakterize özelliklerinden bir tanesi de dönerken stator sargılarında oluşan ters elektromotor kuvvet (zıt EMK) şekli trapezoidaldir veya sinozialdir. Üç fazlı BLDC motorlar; ortalama DC gerilimin güç anahtarları yardımıyla (PWM), belirli iki motor fazının uygun sırayla enerjilendirilmesi yöntemine dayanarak kontrol edilmektedir. Sürülebilmeleri yönünden, DC sürücülerin avantajlarına sahip olmakla birlikte, zorlukları elektronik komutasyon yapılarak elimine edilebilmektedir. Dolayısıyla, bu motorların akım ve torku, gerilim ve hız ile doğrusal ilişkilidir.

BLDC motor statoru, tümleşik halde ince çelik tabaka yığını ve bunun içerdiği yapısal tasarımı gereği boşluklarında bulunan faz sargılarından oluşmaktadır. Fazlar genellikle birbirlerine yıldız bağlı olarak tasarlanır. Fazlar; belirli bir sırayla, sarım sayısıyla ve geometri ile sarılmaktadır. Bu geometrik tasarım ile motor zıt EMK dalga şekli belirlenmektedir. PMSM(Permanent Magnet Syhcronus Machine) diye adlandırılan motorlar da BLDC motorlar gibi senkron motorlardır ve rotor kalıcı mıknatıstır. Aralarındaki temel fark geri EMF voltajlarının şeklidir. Üç statorun eşit bir şekilde sarıldığı BLDC motorlarda ise geri EMF trapezoidaldir(yamuktur).



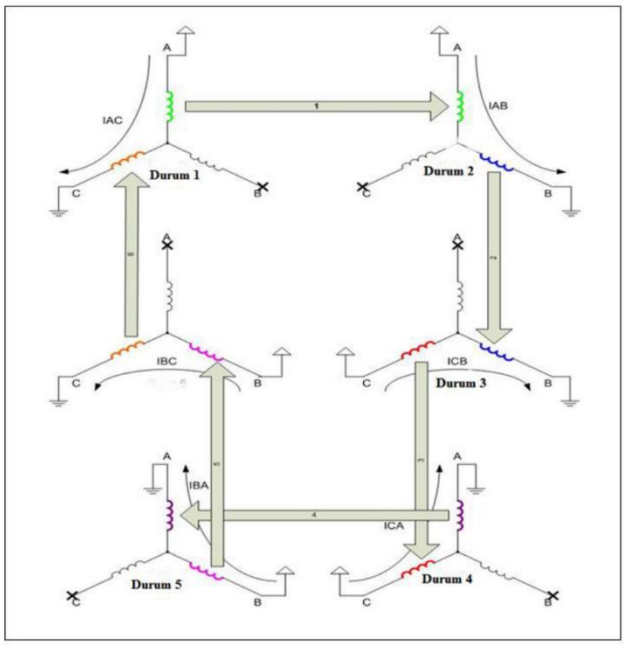
Geri EMF (Zıt EMK) Nedir?

Bobin üzerine akım uygularsak manyetik alan oluşur. Akım taşıyan bir iletken, bir manyetik alan içerisine girdiğinde kuvvete maruz kalır. Yani Elektromanyetik İndiksiyon Teoremi'ne göre iletken yapı, manyetik alanı kestiğinde, EMF iletkende indüklenir. Bu durum mıknatısların bobin sargıları üzerine oluşturmuş olduğu akıdan dolayı meydana gelir ve bobinlerin mıknatıslara göre olan konumunun değişmesinden dolayı değişkenlik gösterir. Uygulanan kuvvetin formülü **F=B.I.L** olarak hesaplanır. F kuvveti, B manyetik alanı, I akımı ve L iletken uzunluğunu simgelemektedir. Oluşan kuvvetin yönü sağ el kuralı ile belirlenir. Orta parmak, baş parmak ve işaret parmağını birbirine dik olacak şekilde ayarlarsak, işaret parmağı manyetik alan yönünü, baş parmak iletkenin hareket yönünü, orta parmak ise iletken üzerinde indüklenen emk'yi temsil eder. Aşağıda gösterilen şekilde sağ el kuralı uygulandığında, indüklenen emk yönünün uygulanan voltajın tersi olduğu görülmektedir. Buna da **geri emf** diyoruz.

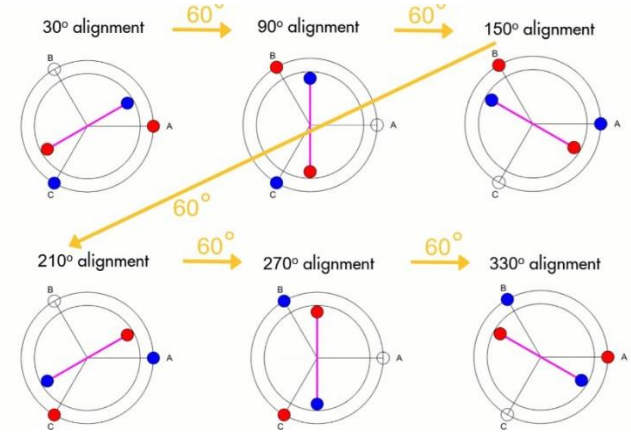


Bu geri emf tespitinin rotor konumunu hissetmede büyük bir katkısı vardır. Rotorun 3 bobinli kısmından birinin pozitif gerilimle diğer bobinin negatif gerilimle beslendiğini söylemek gerekiyor. Yani sağlanan kaynak gerilimi iki bobinden ilerlerken diğer üçüncü kısımdaki bobin ise floating diye adlandırılan boş kısımdır ve buradan motorun geri emf'si okunur.

Motorumuz A, B, C diye adlandırılan üç sargıdan meydana oluşur. Bizim bu sargılardan sadece ikisini enerjilendirmemiz ve belirli yönde akım akıtmamız gerekir. Alttaki şekilde de görüldüğü gibi bu geri emf'lerin konumu belli bir yönde olduğu sürece motorun dönüşü senkronize olacaktır. Bu senkronizasyonu sağlamak için ve ok(-->) ile gösterilen akımın aktığı sargıları da zamanında değiştirerek motorun dönüşünü sağlamak için devrede motor sürüşünde görev alan bu mosfetleri belli bir sırayla anahtarlamak gerekmektedir.

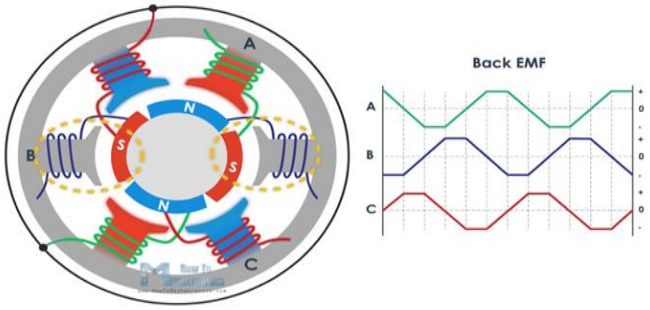


Enerji vermediğimiz sargıdan da az sonra bahsedilecek olan sanal noktanın merkezindeki gerilim ölçülür ve ne zaman sıradaki bobinleri enerjilendirmemiz gerektiğini takip ederiz. Bu şekilde 60 derecelik 6 evre elde etmiş oluruz.

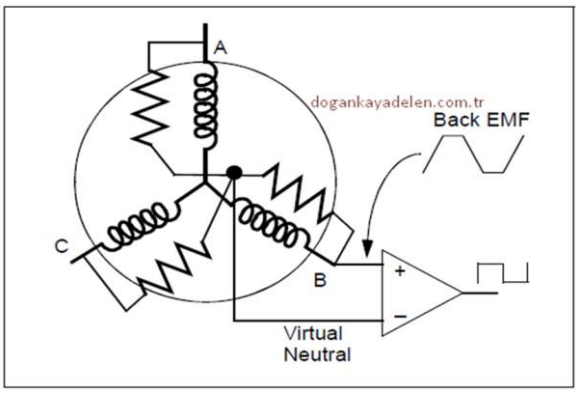


Anahtarlamayı da düzgün yapabilmek için geri emf'yi takip etmek gerekmektedir. Geri emf'yi devrede takip etmek için voltaj bölücü dirençleri kullanmaktayız. Bu dirençlerden birbirine parelel 3 hat kurup hatlardan birini motorun bir fazına diğerini diğer bir fazına olacak şekilde bağlantı

kurduktan sonra bu noktalardaki voltaj düşümlerini kullandığımız mikroişlemcideki analog karşılaştırıcıya göndererek mosfetlerin açılıp kapanması için komut alıyoruz. Faraday İndiksiyon Yasasına göre bunu yapıyoruz. Çünkü mıknatıs bobinin tam ortasına hizalandığında voltaj düşümü olur.

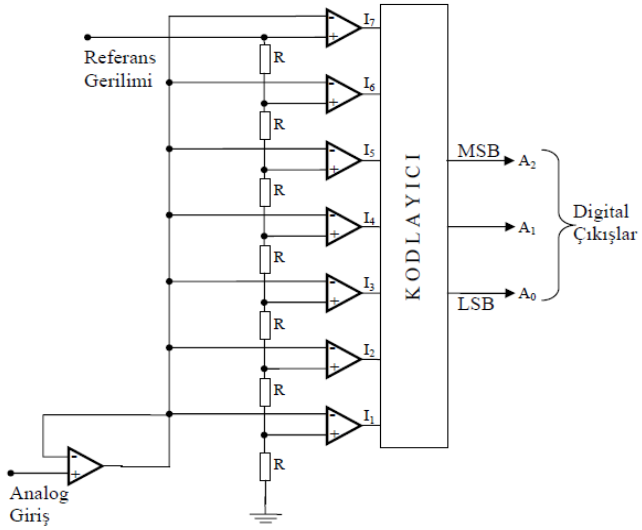


Şekilde görüldüğü gibi üç bobinin tüm voltajlarını birlikte toplarsak bir sonraki geçişe ne zaman geçileceğini tespit edebiliriz. Motorun bobinlerine parelel kurulan dirençlerin birleştiği noktaya "Virtual Neutral Point" denir.



Arduino'daki Analog Karşılaştırıcı:

Analog karşılaştırıcılar aslında gelen analog sinyalini dijital sinyal değerleri olan 1 ve 0 değerlerine çevirmeyi sağlamaktadır. Analog-dijital dönüştürücüler (ADC), genellikle sürekli voltaj veya akım olan analog ölçümlerden elde edilen verileri; hesaplama, veri iletimi, bilgi işleme, depolama ve kontrol sistemlerinde kullanmak için sayısal ifadelere dönüştürür. Sayısal sinyallerin; depolanması, kolaylıkla hata ayıklanması (çeşitli kodlama tekniklerini kullanarak) ve neredeyse gürültüsüz olması bu dönüşümün avantajını ortaya koymaktadır.



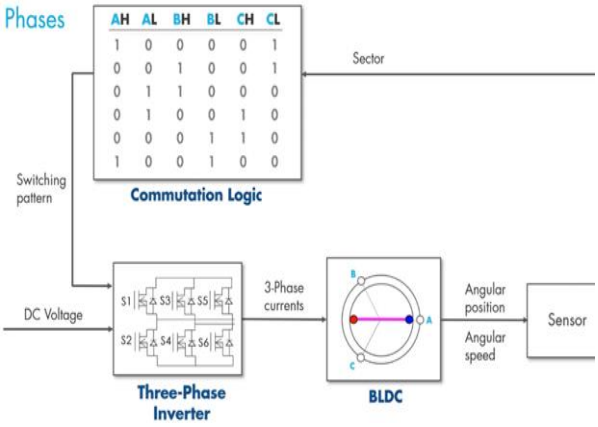
Şekil 1. 3 bitlik paralel ADC

Paralel ADC'nin çalışması oldukça basittir. Analog işaret, analog kaynaktan akım çekilmesini önlemek amacıyla, bir tampon devreden geçirildikten sonra karşılaştırıcıların (-) girişlerine uygulanır. Karşılaştırıcıların (+) girişlerine ise, referans gerilimine bağlı olarak, 1. karşılaştırıcının (+) girişindeki gerilimin tam katları uygulanmaktadır. Her bir karşılaştırıcı için, (-) girişindeki işaret (+) girişindeki işaretten büyükse çıkışı "lojik 0", değilse "lojik 1" seviyesinde olacaktır.

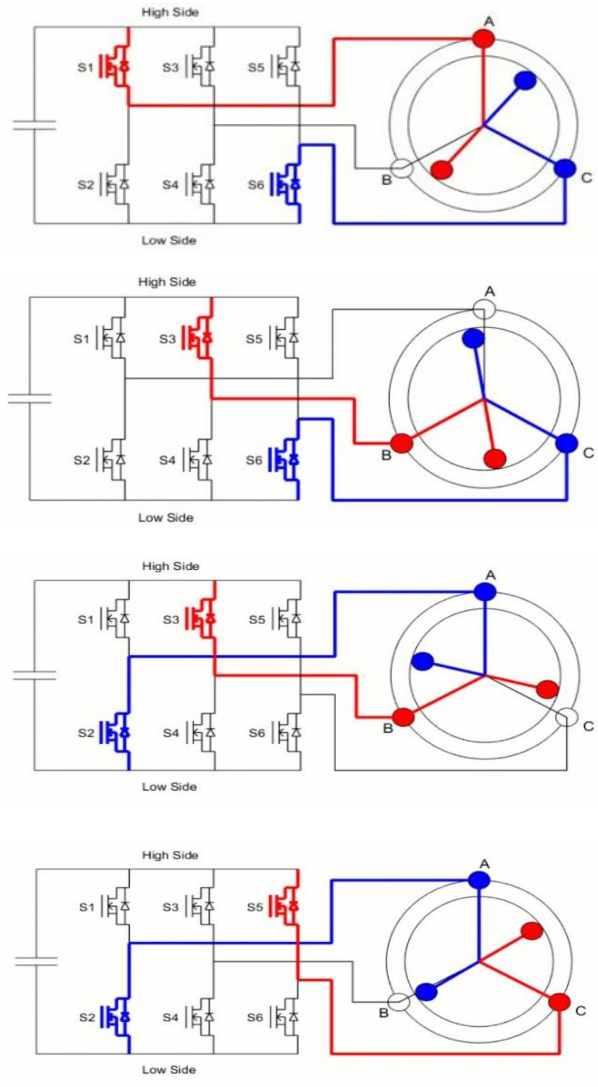
Biz de kendi ESC devremizde mikroişlemci olarak kullandığımız Arduino kartında bulunan Atmega328P'nin karşılaştırıcısını kullanacağız. Bu karşılaştırıcının pozitif girişi AIN0(pin6)'dır. Yani burası "virtual neutral point" kısmıdır. Negatif giriş ise A fazının geri emf'si için pin7(AIN1), B fazının geri emf'si için ADC2 pini iken C fazının geri emf'si için ADC3 pinidir. Karşılaştırıcı her seferinde sanal noktayı bir fazın geri emf'si ile karşılaştırmaktadır. Kodlama kısmında bu karşılaştırıcı sinyallerinden elde edilen sonuca göre PWM çıkışları alınacaktır.

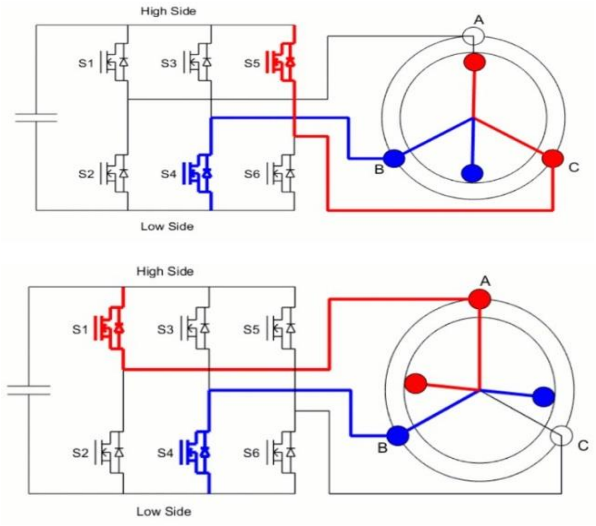
Mosfetleri Düzgün Biçimde Nasıl Anahtarlarız?

Analog karşılaştırıcıya gelen geri emf sinyalleri ile rotor konumu belirlendikten sonra senkronizasyonun sağlanması için uygun mosfetlerin anhtarlanması gerekir. Uygun mosfetlerin anahtarlanmasını geri emf sinyalleri sağlar ve "commutation logic" adı verilen tablo doldurularak uygun mosfetler anahtarlanır.



Şekilde de bu tabloyu görmekteyiz. Mosfetlerin anahtarlamasını incelemek için aşağıdaki görsellere de bakabilirsiniz.



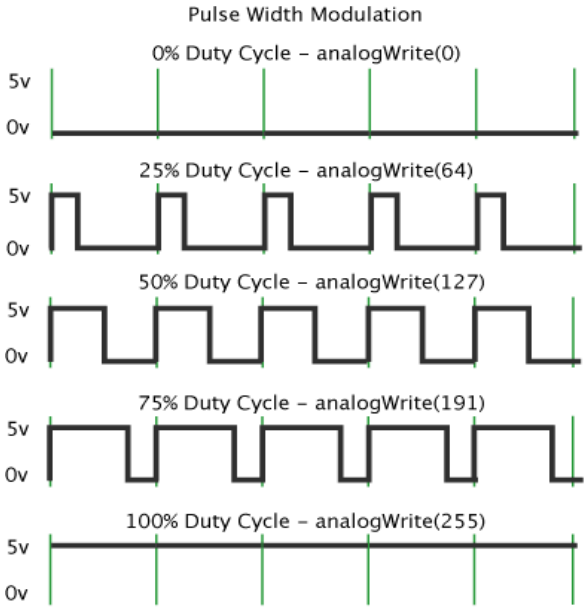


Bu şekiller de sırasıyla iki mosfetin anahtarlamasını göstermektedir. (soldan sağa doğru)

Adımlar arasındaki geçişler motor torku ve dönüşün sürekliliğini sağlarken sargılar üzerine düşürdüğümüz gerilim motorun hızını belirler. Bunu PWM ile yapabiliriz.

PWM Nedir?

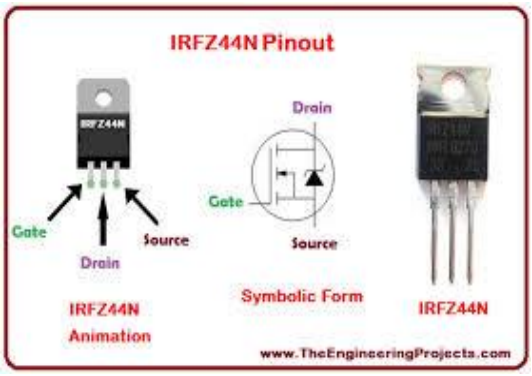
Açılımı **"Pulse Width Modulation"** yani Sinyal Genişlik Modülasyonu olan bu teknik, sinyal işleme veya sinyal aktarma gibi daha çok elektronik devrelerin yanı sıra Arduino veya elektrik makineleri gibi özel uygulama alanlarında da yer alan bir tekniktir. En basit haliyle bir sinyal modülasyon tekniği olarak tanımlanabilir. Sinyal bilgisinin aktarım için uygun hale çevrilmesi amacının yanı sıra güç kontrolü sağlamak ve elektrik makineleri, güneş pili şarj üniteleri gibi özel devrelere destek olmak amacı da taşır. Bu kontrol de tamamen anahtarlama ile sağlanır. Anahtarlama ne kadar hızlı yapılırsa, PWM ile aktarılan sinyalin gücü o kadar da artar. Anahtarlama frekansı çok düşükse, motor ortalama bir voltaj alamayacak ve bu da motor hızlanma ve yavaşlama sürecini belirleyecektir. PWM frekansını makul değere yükselttiğimizde, voltajın ortalaması alınacak ve bu da hız kontrolünü iyileştirecektir. **"Duty Cycle"** yani görev döngüsü kavramı aslında yapılan işlemin periyodunu belirtiyor. Bu döngü düşük seviyelerde ise aktarılan güç düşük olurken, döngünün yüksek seviyelerinde yüksek güç aktarılıyor. Duty Cycle'yi değiştirmeye devam edersek elimizdeki PWM sinyalini sürekli olarak modüle ederiz. Motor hız kontrolüne de bu duty cycle ile oynayarak yaparız.



ESC Devresindeki Malzemeler:

Arduino Uno: Bu parça bizim mikroişlemcimizdir. Burada yapılan kodlamalarla mosfetlerin anahtarlanma sırası belirlenecektir. Ayrıca iki tane buton ile hızlanma ve yavaşlama durumları kontrol edilecektir. Timer fonksiyonlarını kullanarak istediğimiz düzeyde PWM frekansı üreterek dönüş hızını kontrol ediyoruz.

Mosfetler: Yüksek hızda anahtarlama yaptıkları için ve güç kaybı yaşatmadığı için tercih edilmişlerdir. Benim devremde **IRFZ44N** mosfetler kullanılacaktır.



Mosfet Entegreleri: Devreyi sürebilmek yani, mosfetleri sürebilmek kolay bir yol değildir. Bu durumda mosfeti Arduino gibi herhangi bir mikroişlemci çıkışına direk takmak doğru olmayacaktır. PWM frekansı yükseldikçe mosfetin giriş kapasitesini hızlı bir şekilde doldurup boşaltmamız gerekir. Bu yüzden bu konuda mosfet sürücü entegreleri daha yardımcı olacaktır. Arduino çıkışı 3-5 miliamper akım sağlayacağı için yeterli değildir. Bu yüzden akımı yükseltecek olan entegreleri tercih ettik. Benim devremde olacak olan entegre ise **IR2104**'dür. Aynı zamanda bu entegreler devredeki tetikleme sinyallerinin daha dik bir şekilde gelmesini sağlayacaktır. Aksi takdirde tetikleme sinyalleri dik olmazsa mosfetler gereksiz yere ısınabilmektedir. Bununla birlikte böyle entegreler devredeki parazitik etkileri en az düzeye indirir.

IOR Rectifier — **IR2104(S)&(PbF)**

HALF-BRIDGE DRIVER

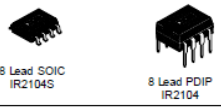
Features

- Floating channel designed for bootstrap operation
  Fully operational to +600V
  Tolerant to negative transient voltage
  dV/dt immune
- Gate drive supply range from 10 to 20V
- Undervoltage lockout
- 3.3V, 5V and 15V input logic compatible
- Cross-conduction prevention logic
- Internally set deadtime
- High side output in phase with input
- Shut down input turns off both channels
- Matched propagation delay for both channels
- Also available LEAD-FREE

Product Summary

V_{OFFSET}	600V max.
$I_{O+/-}$	130 mA / 270 mA
V_{OUT}	10 - 20V
$t_{on/off}$ (typ.)	680 & 150 ns
Deadtime (typ.)	520 ns

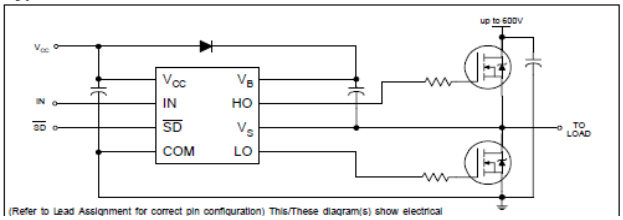
Packages



Description

The IR2104(S) are high voltage, high speed power MOSFET and IGBT drivers with dependent high and low side referenced output channels. Proprietary HVIC and latch immune CMOS technologies enable ruggedized monolithic construction. The logic input is compatible with standard CMOS or LSTTL output, down to 3.3V logic. The output drivers feature a high pulse current buffer stage designed for minimum driver cross-conduction. The floating channel can be used to drive an N-channel power MOSFET or IGBT in the high side configuration which operates from 10 to 600 volts.

Typical Connection



(Refer to Lead Assignment for correct pin configuration) This/These diagram(s) show electrical

Dirençler: Mosfetin gate kısmına koyduğum dirençler akım sınırlaması yapması için tercih edilmiştir. Diğer dirençler ise daha önce de belirttiğim gibi voltaj bölücülüğü sayesinde geri emf tespitinde kullanılıp analog karşılaştırıcıya sinyal gönderilmesini sağlar.

Diyotlar ve Kapasitörler: 10 uF'lik kapasitörler entegremize girecek olan DC voltajın dalgalanmalarını önleyecekken, **2.2 uF**'lik kondansatör ve **1N4148** diyot ile **"Bootstrap Tekniği"** denen durumu kullanacağız. 2.2 uF değerindeki kapasitöre **bootstrap kapasitörü** diyoruz. Her entegreye bağlı iki mosfet kendi ilgili olduğu bootstrap kapasitörünü şarj için kullanarak, gate voltajının source voltajından bootstap kapasitörünün değeri kadar fazla olmasını sağlar. Bootstrap kapasitörü voltajı, VCC'den diyot voltajını çıkardığımızda elde edilen **VB** değeridir. Diyodumuz ve bootstrap kapasitörü VCC üzerinden, **LO** pinine bağlı olan mosfet her **ON** olduğu zaman, şarj olmasını sağlar. Benim kuracağım devrede bootstrap kapasitörleri entegremizin sadece **LO** uçlarına bağlı olan mosfetler **ON** olduğu zaman uçlarına +12 Volt uygulanmaktadır. Bu sayede kapasitörler şarj olabilmektedir. Şarj olan bu kapasitörlerin voltajı

üstlerinde kalan **HO**'ya bağlı mosfetleri açmak yani **ON** yapabilmek için kullanılır. **HIGH** taraftaki mosfetler her **ON** olacağı zaman source voltajına bu kapasitörlerdeki voltaj eklenir, böylece gate voltajı her zaman source voltajından kapasitör voltajı kadar fazla olmuş olur. **HIGH** taraf mosfetin source bacağı floating olur yani toprağa göre referans alınmayan bir nokta olur ki bu da zaten 3 fazlı motorun bir fazına bağlanacaktır.

Bootstrap Tekniği Niye Kullanılır?

Aşağıdaki resimde bu tekniğin bir görselini paylaştım. Mosfetleri ON yapabilmek için **Vgs** voltajı uygulamanız gerekmektedir fakat bu voltaj her zaman mosfetin source bacağına göre **+ voltaj** olmalıdır. Q2 nin source bacağı, besleme ile aynı referans noktasını paylaştığı için Q2 mosfet driver olmadan bile ON yapılabilir. Fakat Q1 i ON yapmanın pek çok yolu olmakla birlikte biz IR2104 gibi mosfet driver kullanmayı bu yüzden tercih ediyoruz. Mosfet driverda kullanılan bootstrap kapasitör voltajı, mosfeti aktif hale getirmek için kullanılır.

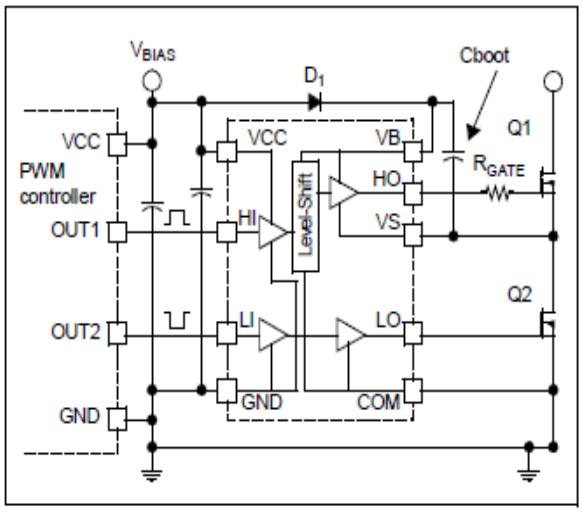


FIGURE 11: *High Voltage Bootstrap Driver IC.*

Fırçasız motor seçimi: Herhangi bir küçük RC araç için tercih edilen bir motor seçilebilir. Ben A2212 serisinden 1400 KV'lik bir motor seçtim. Bu motora 11.1 volt kadar gerilim uygulanacağını varsayarsak 11.1 x 1400 = 15540 rpm(Revolutions Per Minute) yapmasını beklemekteyiz(maks). RPM'i bir dakikadaki dönme sayısı olarak ifade edilebiliriz. Dönüş hareketi yapan fiziksel sistemlerin dönme hızını veya frekansını ifade eder.

Güç kaynağı seçimi: Genellikle üç fazlı fırçasız motorların dönüşü için 3S Li-Po piller seçilmelidir. S değeri 3.7 Volt'a karşılık gelir ve 3S deyince de 12 Volt'a yani devremizin çalışması için gerekli Vcc değerine ulaşmaktayız. C değeri pilin kapasitesidir. Anlık olarak çekebileceğimiz akımın kapasitesidir. Mesela 2200 mAh'lık 3S ve 30C'lik bir pili düşünürsek (2200 x 30)/1000 = 66 Amper anlık akım çekme gözlemleriz.

Fırçalı motorlar, DC voltajı ile kolayca kontrol edilirken BLDC motorda DC gücünü üç fazlı AC gücüne dönüştürmek için karmaşık bir denetleyici gereklidir. Önemli kavramları okuduktan ve devre için kullanılacak malzemeleri seçtikten sonra ESC yapımına geçebiliriz. ESC tasarlarken bazı hususlara dikkat etmek gerekiyor.

1) Motor kontrol topolojisi ve motor tipi
2) ESC verimliliği ile uçuş süresi ve maliyet karşılaştırması
3) Maksimum uçuş hızı parametreleri 12000 rpm'ye kadar veya 1kHz'den yüksekse daha fazla rpm'ye kadar çıkabilmelidir.
4) EMC şartları sağlanmalıdır.

Tasarlamak istediğimiz ESC devresi 30 kHz ile 60 kHz arasında olmalıdır. Çünkü yüksek hızlı motorlar ve sensörlerimiz için düşük empedans sağlamalıyız. Bunu da kontrol algoritmasını yazarken ayarlayabilmemiz gerekmektedir.

Arduino İnterrupt:

İnterruptlar yani kesmeler en temel anlamda işlemcinin hali hazırdaki aktivitesini kesip başka bir iş yapabilmemize imkan sağlar. Kesme nerede gerçekleşirse gerçekleşsin programın herhangi bir anında kesmeye gidilir. Kesmedeki işlem bittiğinde nerede kesmeye gidilmişse oraya tekrar dönülür. İkiye ayrılır.

1) Dış Kesme(External İnterrupt)
2) Zaman Kesmesi (Timer İnterrupt)

Arduino Uno iki adet donanım(dış kesme) interruptına sahiptir. Gerekli ayarlamalar yapıldıktan sonra pin2 ve pin3'deki voltaj değişikliklerine bağlı olarak kesme oluşmaktadır. Dış kesme dört farklı modda yapılabilmektedir.

1) **LOW:** Dijital pindeki gerilim 0 olduğunda kesmeye girer.
2) **CHANGE:** Belirli dijital pinde oluşacak her gerilim değişiminde kesmeye girer. Yani pindeki gerilim 0'dan 5'e yükseldiğinde veya 5'den 0'a düştüğünde kesmeye girer.
3) **RISING:** Yükselen kenar olduğunda kesmeye girer. Yani dijital pindeki 0'dan 5 Volt'a çıktığında kesmeye girer.
4) **FALLING:** Düşen kenar olduğunda kesmeye girer. Yani dijital pindeki gerilim 5'den 0'a düştüğünde kesmeye girer.

Dış Kesme: Bu modlardan herhangi birisi kullanılarak istenen donanım interrupt'ının tetiklenmesi sağlanır.

attachInterrupt(digitalPin, kesmefonksiyonu, mod);

dijitalPin = 0 veya 1 girilebilir. 0=INT0=PIN2, 1=INT1=PIN3

Kesme Fonksiyonu: Kesme fonksiyonun adı "**void** ledyak()" gibi bir şey olabilir. Böyle fonksiyonları "**void** loop()" fonksiyonundan sonra yazmalıyız.

Mod: Yukarıda bahsettiğim dört farklı moddan bir tanesi girilir.

Zaman Kesmesi: Bu kesmede istediğimiz süre sonunda istediğimiz süre sonunda, istediğimiz işlemleri yapmaya yarayan kesmedir. Örneğin her 10 saniyede bir sensörden veri okumayı istersek bu kesmeyi kullanabiliriz. Arduino Uno'da iki adet timer birimi vardır. Bunlar Timer0 ve Timer1. Timer0 8 bit, Timer1 ise 16 bittir. Bu nedenle hafıza bakımından düşünüldüğünde Timer1 ile çalışmak daha mantıklıdır.
Timer fonksiyonları kodumuzun ilk başında olmalıdır ve şu komutlar konmalıdır.

KULLANDIĞIMIZ KODLAR

```
cli();                          //interreptlar durduruluyor
TCCR1A = 0;                     // TCCR1A register 0'lanıyor
TCCR1B = 0;                     // TCCR1B register 0'lanıyor
TCNT1=0;                        // Sayac Değerini 0'la-----Timer1'in kendi saydığı değer. süre
OCR1A = 15624;                  // karşılaştırma registerina değer atanıyor. Süre hesaplamasından.
TCCR1B |= (1 << WGM12);         // Mod ayarlama CTC modu aktif. WGM12 bit'ini lojik '1' yaptık.
TCCR1B |= (1 << CS12) | (1 << CS10); // Prescaler değeri ayarlanıyor. 1/1024 olarak ayarlandı. CS12 ve
                                //CS10 bitlerini lojik '1' yaptık
TIMSK1 |= (1 << OCIE1A);        // sayac değeri ve bayrak sıfırlandı. Sayma işlemi için gerekli
sei();                          //interreuptlar aktif

ISR(TIMER1_COMPA_vect)
{
 //timer1 interrupt ı bu kısma Timer'dayken ne yapılması gerektiği yazılacaktır. void loop'tan önce gelir
}
```

Yine Timer1 configurasyonunun pinleri vardır. Bunlardan son 3 biti prescalar oranını vermektedir.

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ICNC1: Input Capture Noise Canceler**

$$OCR1A = \frac{Osilator\ Hızı\ (16MHz=16x10^6)}{Gecikme\ Frekansı(Hz)*Prescalar\ orani} - 1$$

$$Gecikme\ Frekansı = \frac{1}{T(saniye)}$$

OCR0A= Timer0'in alacağı değer. Max 255
OCR1A= Timer1'in alacağı değer. Max 65535

Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

TCCR1B |= (1<<CS12) | (CS<<10) yazarak prescalar oranını 1024 yaparız. Zaman Timer'ini 3 modda kullanabiliriz. Bunlar normal mod, CTC(Clear Time on Compare) ve PWM modudur. Biz CTC yani karşılaştırma Timer'i kullanacağız.

TCCR1B |= (1<<WGM12) komutuyla CTC modu aktif olur. Hesaplanan OCR1A registeri CTC

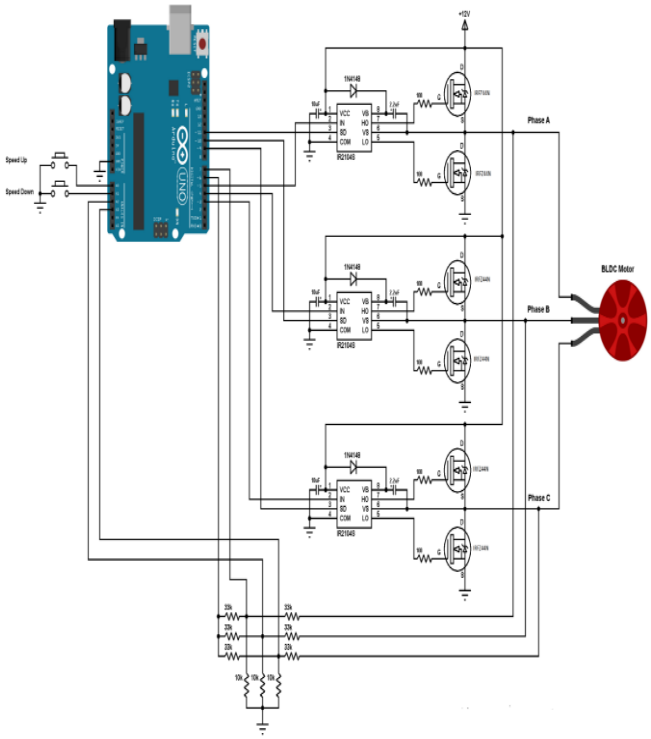
modunda çalışan timer'in set pointini tutar. Proses value yani TCNT1 registerinin tuttuğu sayaç değeri OCR1A ile karşılaştırılır. Bu karşılaştırma bizim belirlediğimiz değerle aynı olursa timer tetiklenir. Bunun için de TIMSK1 registerinin OCIE1A biti 1 yapılır. Böylece ISR(Interrupt Service Rutin-interrupt) fonksiyonun çalışması için gerekli bit ayarlanmış olur.

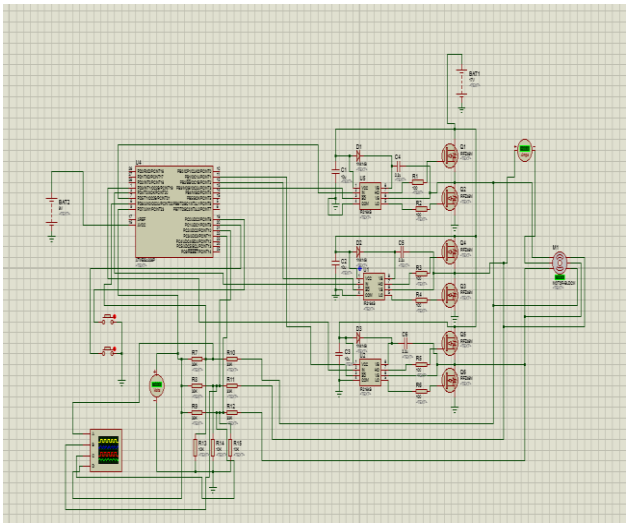
TIMSK |= (1<<OCIE1A)

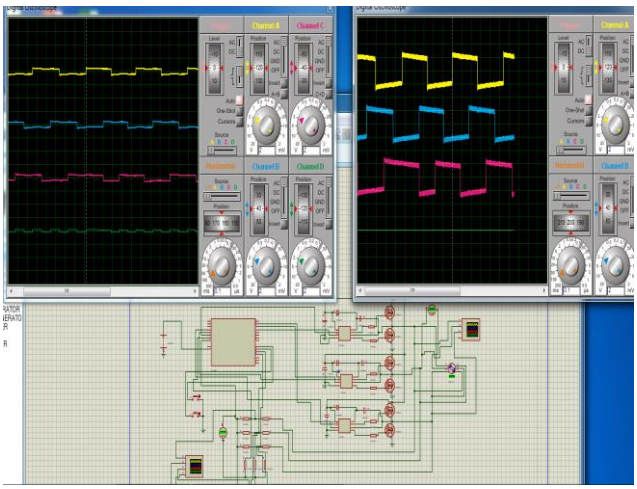
Bu arada bizim ayarlamak istediğimiz PWM çalışma frekansı 31-32 kHz civarında olacaktır.

IV. ESC Devresinin Benzetim Aşaması:

Bu teorik bilgileri de öğrendikten sonra Proteus benzetim programını kullanıyoruz. Arduino kütüphanelerinin yetersizliğinden ötürü Atmega328P işlemcisini kullandım. Buraya Arduino programında derlediğim kodun .hex uzantılı halini yükleyerek benzetimi gerçekleştiriyorum. Ayrıca bir de devrenin diyagramını da burada paylaşıyorum.

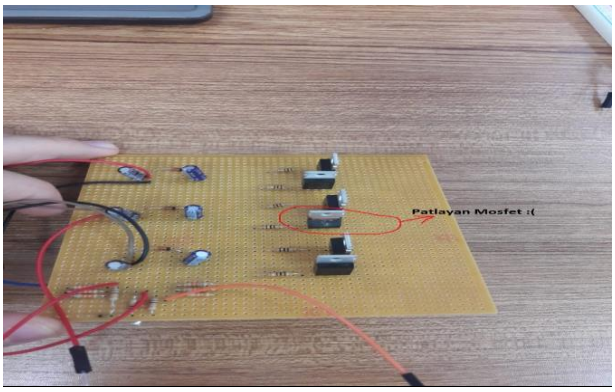


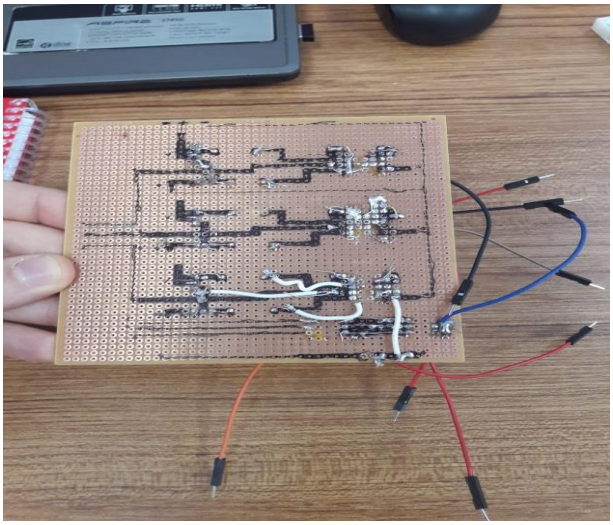




V. ESC Devremizin Yapılması

Malzemelerimizi delikli board veya **pertinaks** denen ince malzemeye yerleştirebiliriz. 15cmx15cm ebatlarında bir parça bize yetecektir. Buraya yapmamızdaki en büyük sebep yüksek akım ile çalışmanın jumper kablolar ile mümkün olmamasıdır. Bu yüzden lehim yolları ve kalın kablolar kullanarak tasarımı gerçekleştiriyoruz.





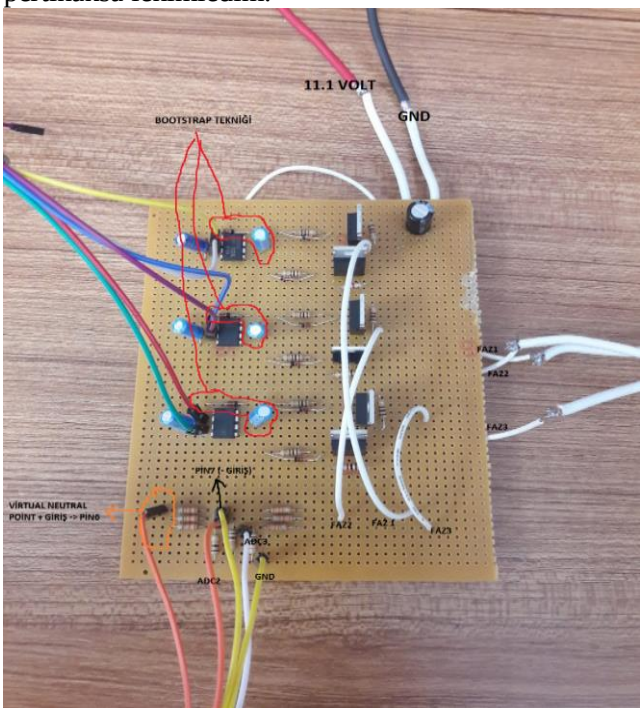
Arduino'ya da kodumu yükleyip devreyi çalıştırmaya gelince, üstten bakınca dördüncü alttan bakınca üçüncü mosfetin bir anda patladığını gözlemledik. Evet, mosfet patladı!

Mosfetlerin neden patlayabileceği üzerinde araştırmalar yaptım. Böyle bir sıkıntının yaşanmasında dört farklı sebep olduğunu düşünüyorum.

1) Devremizin GND bağlantısı tamamlanmamış olabilir.
2) Devreye güç kaynağı olarak vermek istediğimiz Li-Po bataryanın mAh değerinin 2200 olması ilk anda fazla akım girmesine neden olmuş ve bunun da mosfeti patlatabilme ihtimalinin olması. Çünkü biliyoruz ki 2200 mAh'ı 1000'e bölersek 2.2 Amper akım sağladığını görüyoruz. Bu da bizim mA düzeylerinde çalışan mosfet entegrelerini olumsuz etkilemiş olabilir.
3) Devrenin girişine ebatları biraz büyük olan bir kapasitör yerleştirmek gelen DC voltajın daha az dalgalanma yaparak devreye girişini sağlayacaktır. Bizim ilk devremizde böyle bir kapasitör yoktu.
4) Mosfetlerin gate ve source arası direnç olmalıdır. Aksi takdirde gate gerilimi uygulanmasa bile anahtar görevi gören mosfet iletimde olacaktır yani yanlış yere tetiklenecektir. Anahtarlama elemanları kesime girme esnasında anahtar uçlarında hızlı bir gerilim yükselmesine ve iletime girme esnasında hızlı bir akım yükselmesine maruz kalırlar. İşte direnci bu yüzden, bir daha patlama olmaması maksadıyla yerleştirdim.

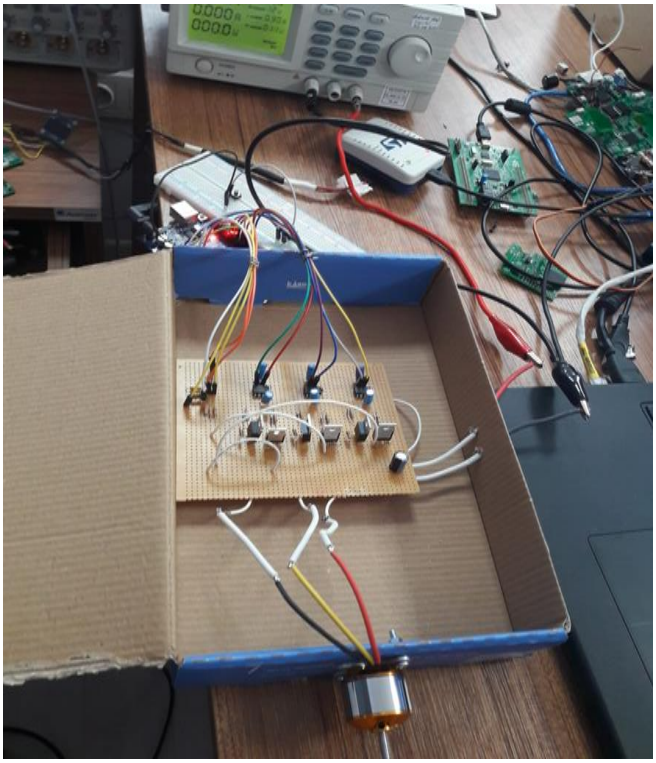
En kuvvetli sebebin bu dördüncü madde olduğunu düşünmekteyim. Çünkü internetten araştırdığım kadarıyla böyle bir sorunla yani gate source arası direnç takmayıp olumsuz sonuçlar alan ilk kişi ben değilmişim. Normalde teorik olarak böyle bir durumla karşılaşmadım ama mosfetlerin gate kısmının havadaki elektronlardan bile etkilenip kontrolümüz olmadan dolabileceğini öğrendim. Bu yüzden de her mosfetin gate ve source bacakları arasına 1k direnç takmayı uygun gördüm.

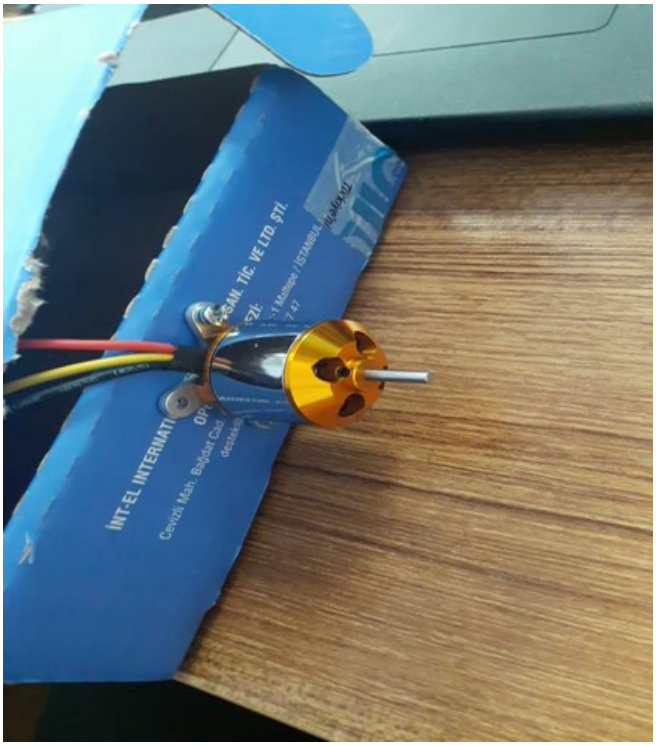
Bunlara ek olarak gate dirençlerine parelel olarak ters yönde olmak şartıyla yine 1N4148 diyotlarını taktım. Bunları koymamın sebebi, gate direncinin üzerinden atlayarak mosfetlerin gate kapasitanslarını hızlı bir şekilde boşaltması ve kapanma süresini azaltmasıdır. Başka alınması gereken tedbir görmedim ve malzemelerimin modellerini değiştirmeden devremi yeniden pertinaksa lehimledim.



Devrenin bir önceki gibi sıkıntı yaşatmaması için yani patlayan mosfetin insanların gözüne gelmemesi için bu pertinaksa ve motora uygun dandik bir kutuya koymaya karar verdim. Bu mosfetler patlasa bile kimsenin gözüne zarar vermemelidir.

Bu doğrultuda ESC devremi bir kutuya yerleştirdim, işyerinden en kısa sürede bir güç kaynağı ayarladım.





Kutulu devreyi güç kaynağının yardımıyla sürmeye çalıştım. Biraz dönüşünde zorlansa da motor dönebilmektedir. Bunun nedenlerini araştırmakla birlikte geri emf yakalamaktaki zorluktan kaynaklandığını düşünmekteyim. Ancak yine dönme yaparak beni gayet mutlu etmiştir.

VI. Sonuçlar:

Bu makalede fırçasız DC motorların çalışma prensibini ve ardından bu motorları kontrol etmeye yardımcı olan ESC devrelerinin çalışma mantığını anlatmaya çalıştım. Ardından bir ESC devresi tasarlamaya çalıştım. Mikrodenetleyiciye yazdığım kodlarla sırasıyla geri emf sinyallerini takip etmeye çalıştım. Aynı zamanda motorun çalışabilmesi için uygun olan 30kHz ile 50 kHz arası bir değeri yakalamayı çalıştım.

VII. Kaynakça:

[1]: Motor-control considerations for electronic speed control in drones (By Kristen N. Mogensen System Engineer, Industrial Systems and Motor Drives)

[2]: Disturbance Observer Structure Applied to Sensorless Brushless DC Motors Drive Samuel Wang, Chin-Hsuan Lu, and An-Chen Lee

[3]: Güç Elektroniği ve Motor Sürücü Devrelerinde Verim Artırma Yöntemleri (Ahmet Berkant ECEVİT)

[4]: Bir Fırçasız Doğru Akım Motoru İçin Sensörsüz Konum Algılama Yönteminin Belirlenmesi ve Gerçeklenmesi (Fatih SEREZ)

[5]: PWM İşaretinin Mikrodenetleyici ile Üretimi (Otomasyon Dergisi)

[6]: ATmega48A/PA/88A/PA/168A/PA/328/P (Microchip)

[7]: MathWorks Motor Control (Melda Ulusoy)