OĞUZHAN ÇİFTÇİ

16.10.2016

DENEYLERLE ELEKTRONİK
-Arduino Destekli-

Oğuzhan Çiftçi
Deneylerle Elektronik
İstanbul-2016
ISBN

Editör

Musahhih
Osman Kibar
Dizgi

Mizanpaj

Kapak

Baskı

ÖNSÖZ-I

Arduino, genellikle hobi amaçlı elektronik devre ve deneylerde tercih edilmektedir. Arduino, aynı zamanda başlangıç aşamasında, meraklılarına büyük kolaylık sağlamaktadır. Öğrencimiz Oğuzhan Çiftçi tarafından hazırlanan bu kitap Arduino'ya yeni başlayanlar için güzel bir referans niteliğindedir.

Kitapta elektronik deneylere de yer verilmiştir. Her deney için elektronik devre eleman ve ekipmanlarının özet bilgilerine yer verilerek, hazırlanan örnek deneyler için ön bilgi mahiyetinde açıklamalar yapılmıştır. Ayrıca algoritma ve kod yazma ile ilgili de yapılan açıklamalarla, programlamaya giriş konusunda bilgilere yer verilmiştir.

Öğrencimiz Oğuzhan'ı yazdığı bu kitap için tebrik ediyor ve yapacağı diğer çalışmalar için de kendisine başarılar diliyorum.

Prof. Dr. Cem Ünsalan
Marmara Üniversitesi
Mühendislik Fakültesi
Elektrik ve Elektronik Mühendisliği
Bölüm Başkanı

ÖNSÖZ-II

Bugün gerek yardımcı kaynaklarının çokluğu, gerek ulaşılabilirliğinin kolay olması sebebiyle Arduino dünya çapında büyük bir üne sahiptir. Arduino ile ilgili örnek kod ile deney şematiklerinin ve temel elektronik bilgilerinin derlenip toplandığı Deneylerle Elektronik kitabı, her yaştan Arduino ve elektronik meraklısına hitap eden ve başlangıç aşamasında elektronik adına bilgilerin kısa ve öz olarak açıklandığı bir başucu kaynağı niteliğinde hazırlanmıştır.

Oğuzhan'ı emek verdiği bu kitap için tebrik ediyor ve yapacağı tüm diğer çalışmalarında da kendisine başarılar diliyorum.

Doç. Dr. Emre ARSLAN
Marmara Üniversitesi
Mühendislik Fakültesi
Elektrik ve Elektronik Mühendisliği
Elektronik Anabilim Dalı

MUKADDİME

Deneylerle Elektronik kitabı, büyük bir derya olan elektroniğe giriş aşamasında, herkese yardımcı olabilecek şekilde hazırlanan ve yine robotik çalışmaların vazgeçilmezi Arduino ile desteklenen deneylerin bulunduğu bir kitap. Bu kitap öncelikle elektronik devre elemanları hakkında genel bilgileri bulabileceğiniz bir başucu kitabı mahiyetinde değerlendirilebilir. Yazarken ki öncelikli amacım elektroniğe ve robotiğe merak salmış, öğrenme yolunda yeni adımlar atmaya başlayanlara, elektroniği sevdirmek ve aslında bunun hakkında merak uyandırmaktı. Bu sebeple, kitapta elektronik adına her şeyi ansiklopedik bilgilerle yazmak yerine, kısa bilgiler halinde özet geçmeye gayret gösterdim. Böylece kimi zaman sıkıcı olabilecek ve aslında başlangıç seviyesinde çok da fazla ihtiyaç duyulmayan bilgileri kısmış oldum. Bunu yapmamın bir diğer amacı da, aslında her bir elektronik devre elemanın özelliğinin, burada bahsedilen ortak özellikleri haricinde, çok yüksek oranda değişiklik gösterebilmesi. Bu tür geniş kapsamlı bilgileri, yapılması planlanan devrelerde, önceden, her bir malzemenin 'datasheet' lerine bakılarak öğrenmek daha isabetli olacaktır.

Kitabın ilk bölümünde elektronikte sıkça kullanılan malzemelerin tanıtımından sonra ise, 12 örnek deneyle pekiştirme yapılması planlandı. Örneklerin çoğunluğu, bugün tüm dünyada en çok ilgi gören ve hem yardımcı kaynaklarının çokluğu hem de erişim kolaylığıyla, elektroniğe başlayan neredeyse herkesin girdiği ilk kapı sayılabilecek, Arduino destekli olarak yürütüldü. Bu sayede, kendinizi geliştirme hızınız artacak ve büyük bir vakit kaybından da kurtulmuş olacaksınız. Örnek

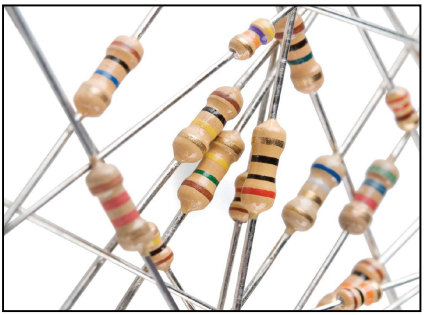
deneyler basitten zora doğru sıralanmış olup seviyenizi devamlı artırmaya yönelik seçildi.
Deneylerle Elektronik kitabında tüm deneyler için kullanılması planlanan, toplu malzeme listesi sayesinde de gerekli malzemeleri kolayca temin edebileceksiniz. Bu liste olabildiğince optimum düzeyde tutulmaya çalışıldı. Ayrıca bu süreçte size yardımcı olacak kaynak sitelerin linklerini de kitabın sonunda "Faydalı Linkler" kısmında bulabilirsiniz.
Bu kitabın hazırlanma sürecinde değerli yardımlarını benden esirgemediği için sayın hocam Doç. Dr. Emre ARSLAN'a ve bana yol gösterip tecrübelerini paylaşan sayın hocam Prof. Dr. Cem ÜNSALAN'a şükranlarımı sunuyorum.
Kitabın tasnifinden büyük emeği olan dayım Osman KİBAR'a da teşekkürü bir borç bilirim.
Ayrıca bu kitabı, desteklerini benden hiçbir zaman esirgemeyen, tecrübeleriyle bana hep yol gösteren anneme ve babama ithaf ediyorum.
Bu kitaptan olabilecek en yüksek seviyede faydalanabilmeniz temennisiyle.

Oğuzhan ÇİFTÇİ
2016

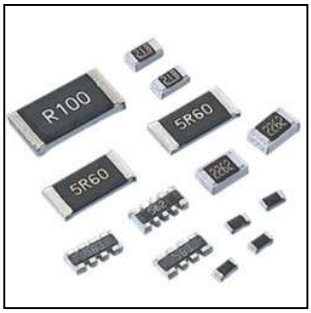
1. TEMEL DEVRE ELEMANLARI

1.1 Direnç Nedir?

Direncin kelime manası, birşeye karşı gösterilen zorluktur. Devre elemanı olan dirençte devrede akıma karşı bir zorluk göstererek birnevi akım sınırlaması yapar. Elektrik enerjisi, direnç üzerinde ısıya dönüşerek harcanır. Direncin birimi ‘‘Ohm’’ ‘dur. Ohm’un ast katları; pikoohm, nanoohm, mikroohm, miliohm, üst katları ise; kiloohm megaohm ve gigaohm’dur.

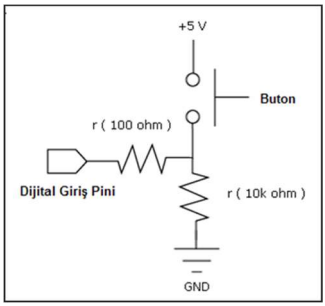


Dirençlerin devrelerdeki kullanım amaçlarından en önemlisi devreden geçen akımı sınırlamaktır. Bu sayede devrelerin stabil olarak çalışması mümkün olur ve hassas devre elemanlarının yüksek akımdan etkilenip zarar görmesi engellenmiş olur. Bazı devrelerde dirençler, devredeki gerilimi bölmek için kullanılır. Bu işlem aynı devrenin farklı noktalarında değişik gerilim miktarı ölçülmesi gerektiğinde oldukça kullanışlı olur. Yine bazı devrelerde de ısı enerjisi elde etmek için kullanılırlar. Örneğin inkübasyon yapılan cihazlarda, yüksek watt değerine sahip dirençler tercih edilir. Ancak elektronikte ekseriyetle küçük akım



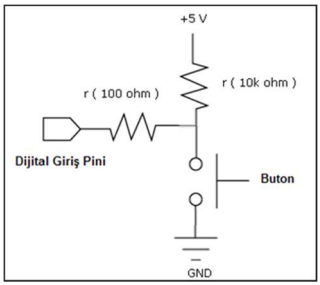
değerleriyle işlemler yapıldığından, tercih edilen dirençlerin değerleri düşük olur.
Bilgisayarlarımızda, telefonlarımızda veya kısaca içinde elektronik devre olan her türlü cihazda kullanılırlar. Ancak bunların da çeşitli boy ve türleri bulunmakta. Örneğin telefonlarımızda bulunan devrelerde SMD (surface mounted device) ve THMD (Through Hole Mounted Device) günümüzde kullanılan çeşitleridir. SMD ismi genel olup bu şekilde devreye yerleştirilen her türlü devre elemanı için kullanılır. Yine de bizim devrelerimizde kullanacağımız devre elemanları bu şekilde olmayıp biraz daha büyük boyutta olacaklar. SMD devre elemanlarının değerleri ekseriyetle üzerlerinde yazılıdır. Ancak bizim kullanacağımız elemanların değerlerine üzerinde, renkler şeklinde kodlanmıştır. Alttaki tablodan her bir rengin karşıladığı sayısal değeri öğrenip elinize aldığınız herhangi bir direncin değerini okuyabilirsiniz.
Arduino ve benzeri mikroişlemci içeren sistemlerde I/O (input/output) pinlerine bir şey bağlanmaması, o pinlerin lojik olarak 0 veya 1 olduğu anlamına gelmez. Pinlerin durumlarından emin olabilmek için bilinmesi gereken iki kavram da Pull-up ve Pull-down dirençleridir.

Pull-down Direnç:

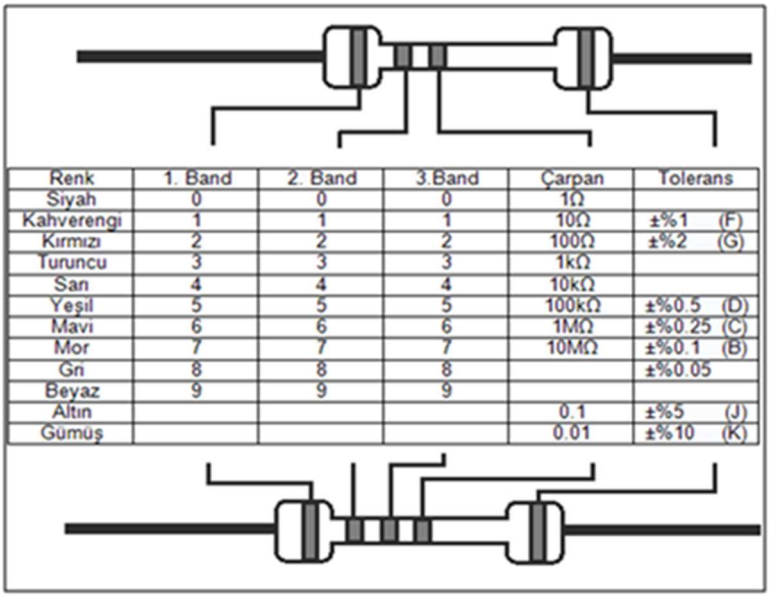


Butona basıldığında 5 Volt arduinonun input ayağına ulaşır. Fakat butondan elimizi kaldırdığımızda arduino-nun pininde 5 volt gerilimi kalır. Bu yüzden de Dijital pin genellikle 4k7 veya 10k ohm luk bir dirençle toprağa bağlanır.

Pull-up Direnç:



Butona basılmadığı durumlarda arduinonun input ayağı 5 volttadır. Butona basıldığında akım arduinonun input ayağı yerine toprağa ulaşmaktadır. Direncin konulma nedeni butona basıldığında 5 Voltun doğrudan toprağa ulaşmasını engellemektir. Genellikle 4k7 veya 10k ohm değerinde direnç kullanılmaktadır.



Renk	1. Band	2. Band	3.Band	Çarpan	Tolerans
Siyah	0	0	0	1Ω	
Kahverengi	1	1	1	10Ω	±%1 (F)
Kırmızı	2	2	2	100Ω	±%2 (G)
Turuncu	3	3	3	1kΩ	
Sarı	4	4	4	10kΩ	
Yeşil	5	5	5	100kΩ	±%0.5 (D)
Mavi	6	6	6	1MΩ	±%0.25 (C)
Mor	7	7	7	10MΩ	±%0.1 (B)
Gri	8	8	8		±%0.05
Beyaz	9	9	9		
Altın				0.1	±%5 (J)
Gümüş				0.01	±%10 (K)

Örnek: Elimizde soldan sağa toplamda 4 çizgisi olan bir direnç olduğunu varsayalım. (Resim 3.3 te üst kısımdaki direnç figürüne benzer). Bu çizgilerin renkleri de sırayla: Turuncu, Siyah, Sarı ve Altın renklerinde olsun. Elimizdeki direncin ohm cinsinden değeri ve toleransı nedir ?

Sonuç: İlk iki rengin tablodaki değerlerini yan yana yazıyoruz:

Turuncu : 3 **Siyah : 0** **=> 30**

Üçüncü rengin tablodaki değeri üsteki sayıya çarpan olarak gelecek:

Sarı : 10kΩ

Yani ;

30 x 10kΩ = 300kΩ

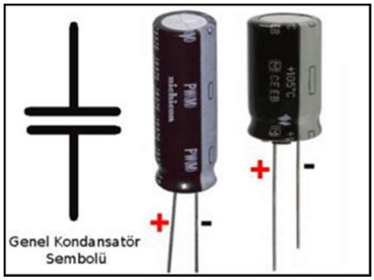
Ve son rengimiz Altın olduğu için toleransımızı da yine tablodaki değeri-mizden alıp yazıyoruz. Sonuç olarak elimizdeki direncin ohm cinsinden değeri:

300kΩ ±%10

olarak kolayca hesaplanabilir.

1.2 Kondansatör Nedir?

Kondansatörler elektrik yüklerini bir süreliğine depo etmeye yarayan devre elemanlarıdır. Diğer bir isimleri de Kapasitördür. Kondansatörlerin sembolü C, birimi ise Faraddır. Kondansatörler yapısal olarak iki iletken levha arasına konulmuş bir yalıtkandan oluşur. İletken levhalar arasında bulanan maddeye elektriği geçirmeyen manasında dielektrik adı verilir.

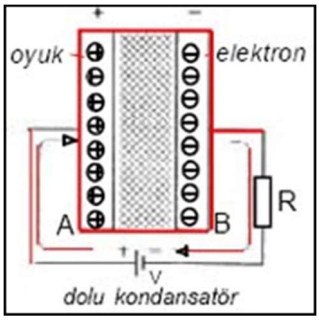


Kondansatörlerde dielektrik madde olarak; mika, kağıt, polyester, metal kağıt, seramik, tantal vb. maddeler kullanılabilir.

Elektrolitik ve tantal kondansatörler kutupludur ve bu nedenle sadece DC ile çalışan devrelerde

kullanılabilirler. Kutupsuz kondansatörler ise DC veya AC devrelerinde kullanılabilir.

Kondansatörlerin elektrik depolama kapasitesi; plakaların yüzey alanına, plakalar arasındaki uzaklığa ve kullanılan dielektrik maddenin cinsine bağlı olarak değişir. Kondansatörler elektriği piller gibi uzun süre depolayamaz, herhangi bir devreye bağlı olmasalar da zamanla boşalırlar.



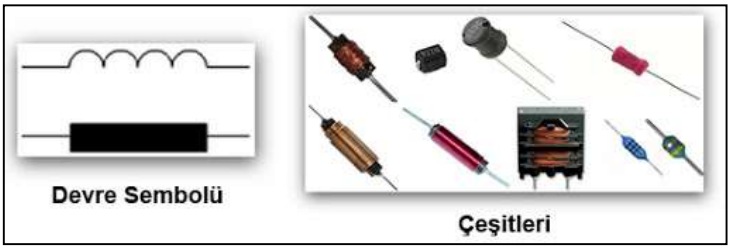
Kondansatörün devre içindeki özelliklerinden birisi yük depolamaktır. Bunun yanısıra yük depolarken içinden akım geçirirken, kapasitesi dolunca içinden akım geçirmez. Böylece tamamen boşalmadan devrenin kendi üzerinden akım geçirmesine imkan vermez. Boşalırken de "+" kutbundan "-" kutbuna doğru devreden akım verir.

Kondansatörün birimi olan Farad çok büyük bir kapasite değerine karşılık geldiğinden, uygulamalarda çoğunlukla faradın pikofarad, nanofarad, mikrofarad, milifarad gibi ast katları kullanılır. Mesela 5 Faradlık bir kapasitör buzdolabı büyüklüğüne kadar ulaşabilir.

1.3 Bobin Nedir?

Bobin bir iletken telin üst üste ya da yanyana sarılması ile üretilen devre elemanıdır. Bobinin birimi Henry (H), simgesi ise L'dir. Bobine AC akım uygulandığında, akımın yönü sürekli değiştiğinden dolayı bobin etrafında bir manyetik alan oluşur. Bu manyetik alan akıma karşı ek bir direnç gösterdiğinden, AC devrelerde

bobinin akıma gösterdiği direnç artar. DC devrelerde ise bobinin akıma karşı gösterdiği direnç, sadece bobinin üretildiği metalden kaynaklanan omik dirençtir. Ayrıca Henry, indüktans değeri bakı-mında çok yüksek bir



değere karşılık geldiği için uygulamalarda çoğunlukla Henry 'nin ast katları kullanılır.

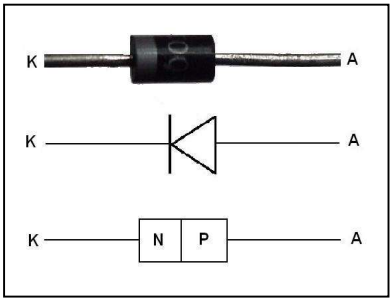
Bobinler ile kondansatörler arasındaki benzerlik her iki devre elemanının da elektrik enerjisini harcamayan reaktif devre elemanları olmalarıdır. Kondansatörlerin elektrik yüklerini depolayabildikleri gibi, bobinler de elektrik enerjisini kısa süreliğine manyetik alan olarak depo ederler. Bu iki devre elemanı arasındaki önemli fark ise; kondansatörler devreye bağlıyken gerilimi geri bırakırken (faz farkı), bobinlerin gerilimi ileri kaydırmasıdır. Bobin ve kondansatörlerin gerilim ve akım arasında oluşturduğu faz farkı uygulamalarda farklı şekillerde fayda ve zararlara neden olur.

Bobinlere akım verilince etraflarında oluşturdukları kuvvete manyetik kuvvet denir. Doğal bir mıknatısı demir tozu içine koyduğumuzda gördüğümüz şekle benzer, etraflarında bir manyetik alan oluştururlar. Bir diğer özellikleri de kondansatör nasıl tam kapasite dolduktan sonra üzerinden akım geçirmeyip açık devre oluşturuyorsa, bobinler de tam tersi biçimde, tam dolana kadar açık devre şeklinde davranırlar.

Bobinlere akım verilince etraflarında oluşturdukları kuvvete manyetik kuvvet denir. Doğal bir mıknatısı demir tozu içine koyduğumuzda gördüğümüz şekle benzer, etraflarında bir manyetik alan oluştururlar. Bir diğer özellikleri de kondansatör nasıl tam kapasite dolduktan sonra üzerinden akım geçirmeyip açık devre oluşturuyorsa, bobinler de tam tersi biçimde, tam dolana kadar açık devre şeklinde davranırlar.
Bobinler ve Kapasitörler de aslında içlerinde bir miktar direnç barındırırlar ve bu direnç (empedans) değerleri kurduğumuz devreleri analiz ederken önemli rol oynar. Ancak bu noktaya kadar öğrendiklerimiz temel oluşturması bakımından yeterli olacaktır.

1.4 Diyot Nedir?

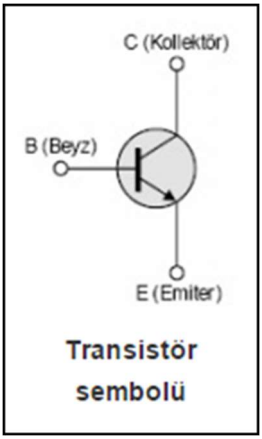
Elektrik akımını tek yönde geçiren devre elemanıdır. Diyotların Anot ve Katot olmak üzere iki ucu vardır. Bunlardan anot pozitif, katot ise negatif ucu temsil eder. Bu nedenle bacakların kuracağımız bütün devrelerde doğru yönde bağlanıp bağlanmadığı çok mühimdir. Diyot, en basit kontrolsüz yarı iletken devre elemanıdır. Geçirme yönünde eşik geriliminin üzerinde küçük değerli bir iç dirence sahip olan bir iletken gibi davranır lar. Tıkama yönünde ise delinme gerilimine kadar, küçük miktarlarda sızıntı akımı geçiren bir yalıtkan gibi davranırlar.



Bu haldeyken üzerlerinden, eser miktar sızıntı hariç, akım geçirmezler.
Doğrultman devrelerinde alternatif akımı doğru akıma (AC-DC) çevirmek için kullanılan devre elemanlarından biridir.

1.5 Transistor Nedir?

Transistör yan yana birleştirilmiş iki PN diyodundan oluşan, girişine uygulanan sinyali yükselterek akım ve gerilim kazancı sağlayan, gerektiğinde anahtarlama elemanı olarak kullanılan yarı iletken bir devre elemanıdır. Transistör kelimesi transfer ve rezistans kelimelerinin birleşiminden doğmuştur.

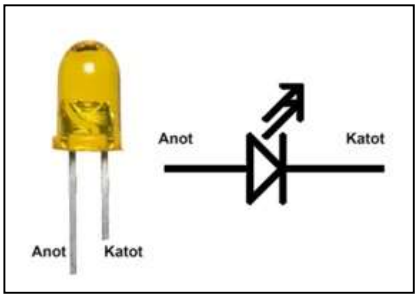


Uygulamada 100000 ‘e yakın çeşidi bulunan ve her geçen gün yeni özelliklerde üretilen transistörler temel olarak bipolar ve unipolar olmak üzere iki gruba ayrılır. Bipolar transistörler NPN ve PNP olmak üzere iki tiptir. Üç kutuplu devre elemanları olan transistörlerin kutupları; Emiter (E), Beyz (B) ve Kollektör (C) olarak adlandırılır. Emiter (yayıcı); akım taşıyıcıların harekete başladığı bölge, Beyz (taban); transistörün çalışmasını etkileyen bölge ve Kollektör (toplayıcı); akım taşıyıcıların toplandığı bölgedir.

1.6 Led Nedir?

Işık yayan diyotlardır. LED (**L**ight **E**mitting **D**iode) kelimesinin kısaltmasıdır. LED’ler elektrik enerjisini ışık

enerjisine çevirmektedir. LED'lerin normal diyotlardan farkı ışık yaymasıdır. LED'ler soğuk ışık yayar, dokunduğunuzda ısınmadığını hissedersiniz. İçerisindeki katkı maddeleri nedeni ile farklı renklerde ışık yayarlar. LED'ler doğru akımla (DC) çalışır. Genellikle 1,6 V - 4 V gerilimle çalışmaktadır. Yarı iletken özelliği gösterir, bu nedenle devreye bağlanırken + ve – uçlarına dikkat etmek gerekmektedir.



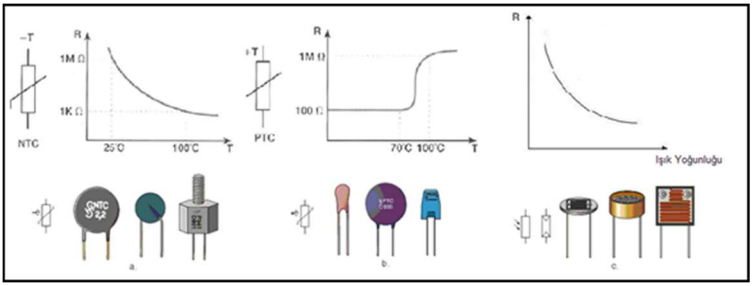
LED'in iki ucu vardır. Bunlardan uzun olanı (+), kısa olanı (-) dir. Ters bağlı olarak 5-10 Volt verildiğinde bozulabilir. Fazla voltaj vermek LED in verdiği ışığın artmasına neden olur, fakat kullanım ömrü de azalır. Aşırı Volt verilmesinde ise patlayabilir. LED'ler genellikle seri bağlanır. Seri bağlı LED'ler 10, 12, 24, 48 Volt doğru akım (DC) veren güç kaynakları veya batarya ile çalıştırılır.

LED'ler isminden de anlaşılacağı üzere bir çeşit diyotturlar. Diyotlar bilindiği üzere akımı yalnızca belirli bir yönde geçirirler (eser miktardaki kaçak akım gözardı edilirse). Bu yüzden + ve - ku-tuplarının voltaj kaynaklarına bağlanma yönleri oldukça önemlidir. Ters bağlanırlarsa Led'ler ışık vermeyeceklerdir. Robotik uygulamalarında oldukça sık kullanılan ledler, programların ilgili yerlerinin çalışıp çalışmadığını fiziksel olarak görebilme imkanı sağlar. Genellikle kontrol amacıyla kullanıldığını söyleyebiliriz. Bizim

yapacağımız deneylerde de öncelikli olarak yine kontrol amacıyla kullanacağız.

1.7 LDR, NTC ve PTC Nedir?

Bunların üçü de aslında temel manada birer dirençtirler. Ancak bazı özellikleri birbirlerinden ayrılmalarına sebep olur. LDR denilen cihazın isminin açılımı **Light Dependent Resistor** olup Türkçe manası Işığa Bağımlı Direnç demektir. Çalışma mantığı ise üstünde bulunan bir malzemenin ışığın şiddetinin artmasıyla LDR nin direnci azalır. Yani ters orantılıdır. Aynı şekilde ışık şiddeti azaldıkça da direnci artacaktır. Aslında temel manada bir sensördür ancak özellikle endüstride tercih edilmez. Bunun yerine küçük elektronik ve robotik deneyler ve uygulamalarda kullanılır.

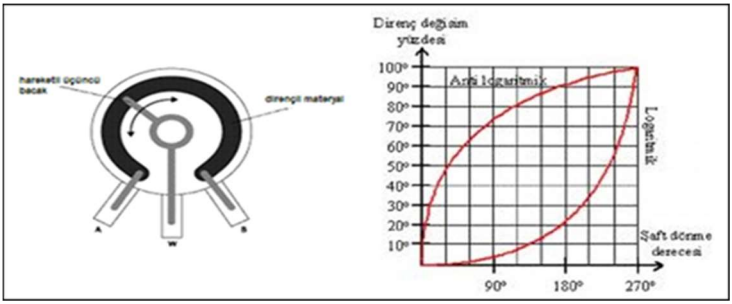


NTC denilen malzeme ise sıcaklığa bağlı olarak direncini değiştiren birnevi sensördür. Açılımı **Negative Temperature Coefficient** olup literatürde Negatif Isı Katsayılı Termistör diye geçer. Çalışma prensibi ise, bulunduğu ortamın veya temas ettiği yüzeyin sıcaklığı arttıkça elektriksel direncinin azalmasıdır.

PTC nin açılımı ise **Positive Temperature Coefficient** olup Pozitif Isı Katsayılı Termistör olarak Türkçeye çevrilebilir. Çalışma prensibi ise NTC nin tersi olarak, bulunduğu ortamın veya temas ettiği yüzeyin sıcaklığı arttıkça elektriksel direncinin de artmasıdır.

1.8 POT Nedir?

Değeri değiştirilebilen bir direnç çeşididir ve asıl ismi Potansiyometre dir. Ancak bu değişikliği kişi manuel olarak elle yapmalıdır. Ayrıca bir analog input eleman olarak kullanılır. Üç bacaklı olurlar ve iç yapısında 2



bacağı sabit, diğer bacağı ise aralarında hareket edebilen bir yapıdadır. Böylece istenilen değerde direnç elde etmek mümkün olmaktadır.

2. MULTİMETRE ve BREADBOARD NEDİR?

2.1 Multimetre Nedir?

Multimetre(Avometre); doğru ve alternatif akım, doğru ve alternatif gerilim ve direnç ölçmeye yarayan ölçme

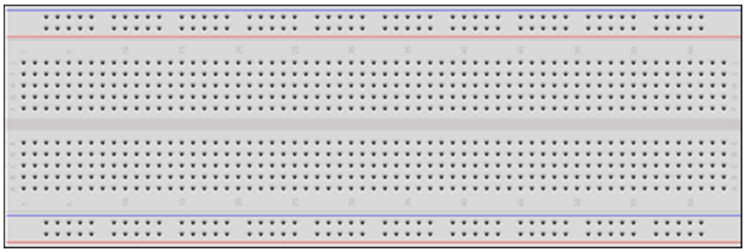
aletidir. Analog ve digital olarak bu-lunan günümüzdeki multimetreler ile direnç dışında frekans, sıcaklık, kapasitans, hfe, duty cycle ölçümü ve transistör, diyot, bobin, transformatör ve sigorta, kısa devre testleri de yapılır. Multimetrelerde ölçüm prob adı verilen ve multimetreye takılan siyah ve kırmızı olmak üzere iki kabloyla yapılmaktadır. Probe adı verilen bu kablolar bakır ve alüminyum gibi iletken malzemelerden yapılır ve dirençleri oldukça küçüktür (Genelde 5 ohm' dan küçük). Ölçüme başlamadan önce bu kabloların sağlamlığı da kontrol edilmelidir. Multimetre buzzer konumuna alınıp kablolar birbirine dokundurulduğu anda ses gel-mektedir. Sesin gelmesi kabloların sağlam olduğunu göstermektedir. Multi-metrede kablolar, ölçülecek olan fonksiyona göre uygun girişlere takılmalıdır. 4 farklı fonksiyonda uç bulunmaktadır ve bunlar **COM, VΩHz (volt, ohm, frekans), mA (miliamper) ve A(Amper)** uçlarıdır.

Yukarıdaki resimde de görüldüğü gibi COM ucu ortak uçtur ve bütün işlemlerde siyah kablo bu uca takılır. Ölçülecek fonksiyona göre kırmızı kablo da diğer

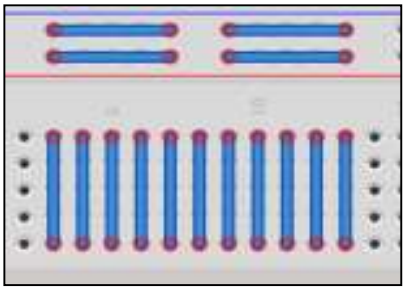
girişlere takılır. Örneğin; direnç ölçümü yapacak isek Ω simgesinin olduğu uca kırmızı kabloyu takmamız gerekmektedir. Ölçeceğimiz akım değeri mA mertebelerinde ise 'mA' ucuna, amper mertebelerinde ise şekildeki **'20A'** ucuna takılmalıdır. Bu multimetre de en büyük ölçülecek değer 20A olarak belirtilmiştir. Multimetrenin kullanımında bu değer üzerinde belirtilir.
Ayrıca dikkat edilmesi gereken bir diğer hususta, ölçüm yapılacak akım ya da gerilimin ölçümünü AC ya da DC olarak yapacağımızı seçmemizdir. AC olarak ölçülecek bir büyüklük DC kademesinde ölçülürse multimetremiz kullanılamaz hale gelebilir ve ciddi sorunlar çıkarabilir.
Multimetreler; ölçüm sonucunu en yakın değerde okumak için mertebelere ayrılmıştır. 200Ω, 2k, 20k, 200k, 2M, 20M vb. gibi. Ölçeceğimiz değer hangi aralıkta bulunmaktaysa o mertebe ayarlanmalıdır. Örneğin; ölçeceğimiz değer 1.5k civarı ise 2k mertebesini ayarlamak bize kolaylık sağlar. 2k mertebesinde 2k 'dan büyük direnç değeri ölçülmez. Bilmediğimiz bir değer için en yüksek mertebeden başlayarak mertebeyi düşüre düşüre doğru aralığı bulabiliriz.

2.2 Breadboard Nedir?

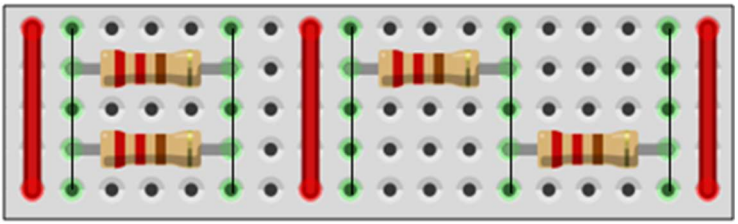
Breadboard devreleri tak çıkar mantığı ile oluşturmamıza yarayan, belli satır ve sütunları kendi aralarında iletken



edilmiş devre tahtasıdır. Ayrıca birden fazla breadboard kendi aralarında çentikleri sayesinde birleştirerek daha büyük devre tahtası elde etmiş oluruz. Breadboard sayesinde devreleri daha hızlı şekilde oluşturarak test etme imkanı elde etmiş oluruz. Bu sayede lehimleme, baskı devre gibi işlemler ile uğraşarak, vakit kaybetmemizi önler.



Üzerinde beşerli satır ve sütunlarla oluşturulmuş devre elemanlarının bacak yuvaları mevcuttur.
Bu satır ve sütunların yandaki görselde nasıl bir bağlantı kurdukları gösterilmiştir. A, B, C, D, E *breadboard*'u yatay olarak tuttuğumuzda dikey sütunlar kendi aralarında birbirleri ile bağlanmıştır. En altta ve en üstte bulunan "+" , "-" olarak belirlenmiş kısımlarda yatay olarak kendi aralarında iletken edilmiştir. Bir zorunluluk olmamakla birlikte mavi çizgili kısma "toprak, - ", kırmızı çizgili kısma "güç, +" bağlanır.



Paralel: Resimde görüldüğü üzere dikey olarak belirtilen ince siyah çizgiler bread-board yuvaları arasındaki bağlantıyı gösterir. İki direnç bu dikey sütunda birbiri ile

paralel bağlanmış olur. Bu olayı birbirlerine lehimleme olarak düşünebiliriz. Bu sayede lehimleme yapılmadan dirençler yuvalara oturtularak paralel bağlantı sağlanmış olur.

Seri: Resimde ince siyah çizgi ile belirtilen dikey sütunlar breadboard yuvaları arasındaki bağlantıyı gösterir. Sol direncin bitiş noktası ile sağ direncin başlangıç noktası aynı dikey sütuna denk getirilmiştir ve bu sayede birbirleri ile bağlantı sağlanıp seri bağlantı oluşturulmuştur.

3. MOTOR ÇEŞİTLERİ

3.1 Servo Motor

Motorlar elektrik enerjisini hareket enerjisine çeviren aygıtlardır. Piyasada bir çok motor çeşidi bulunabilir. Bunlardan bir kaçı DC motorlar, AC motorlar, Step motorlar, Servo Motorlar, RC servo motorlar olarak örnek gösterilebilir. Servo motorlar aslında endüstride en çok kullanılan motor çeşididirler ve çok yüksek enerjilerle çalışıp çok yüksek verimle çok yüksek miktarda hareket enerjisi açığa çıkartabilirler. Servo motorların en büyük avantajları geri bildirim yapabilmeleridir.

Bizim devrelerimizde kullanacağımız servolar ise RC Servo motor olarak isimlendiriliyor ve daha küçük çaplı işlerde kullanılıyor. Ancak ne kadar küçük olarak nitelendirsem de, kibrit kutusu büyüklüğünde bir RC Servo motoru tam verimle kullanabilirsek eğer, kimi modellerinde cm^2 başına 15 kg hatta bazen daha fazla yükleri kaldırabilecek bir enerji sağlayabiliyorlar. Endüstriyel

Servolar da bu sayı tonlara kadar çıkabiliyor. Ayrıca RC Servo motorların bir özellikleri de 0-180 derece arası dönüş yapabilmeleri. Ama bazı özel modellerinde 0-360 derece arasında çalışanları da mevcuttur.

Motor seçimini her motorun kendi seri numarasına göre inceleyebileceğiniz ‘datasheet’ lere bakarak yapıyoruz. Datasheet okumak mühendislikte en mühim şeylerden biridir. Çünkü her malzemenin içyapısını veya bütün özelliklerini bilmek mümkün değildir. O yüzden internetten bir kaç datasheet örneği incelemenizde fayda var.

3.2 Step (Adım) Motor

Açısal konumu adımlar hâlinde değiştiren, çok hassas sinyallerle sürülen motorlara **adım motorları denir**. Adından da anlaşılacağı gibi adım motorları, belirli adımlarla hareket eder. Bu adımlar, motorun sargılarına uygun sinyaller gönderilerek kontrol edilir. Herhangi bir uyartımda motorun yapacağı hareketin ne kadar olacağı motorun adımaçısına bağlıdır. Adım açısı, motorun yapısına bağlı olarak 90° , 45° , 18° , 7.5° , 1.8° veya daha değişik açılarda olabilir. Motora uygulanacak sinyallerin frekansı değiştirilerek motorun hızı kontrol edilebilir. Adım motorlarının dönüş yönü, uygulanan sinyallerin sırası değiştirilerek saati ibresi yönü veya saat ibresinin tersi yönünde olabilir.

Adım motorlarının hangi yöne doğru döneceği, devir sayısı, dönüş hızı gibi değerler mikroişlemci veya bilgisayar yardımı ile kontrol edilebilir. Sonuç olarak adım motorlarının hızı, dönüş yönü ve konumu her zaman bilinmektedir. Bu özelliklerinden dolayı adım motorları çok hassas konum kontrolü istenen yerlerde tercih edilir.

3.3 DC Motor

Robotikte en sık tercih edilen motor tiplerinden biri de DC motorlardır. DC motorlar ucuz, küçük ve etkilidir.

Ayrıca boyut, şekil ve güç bakımından çok çeşitli olmaları da DC motorların sık kullanılmalarının bir diğer sebebidir. DC motorlar robotlarda veya herhangi bir sistemde direkt ya da dişli kutularıyla (redüktörlü ya da redüktörsüz olarak) birlikte kullanılabilirler. DC motorlara bir güç kaynağı bağlandığında DC motorun dönüş yönü akımın yönüne bağlıdır. Akımın yönü terslendiğinde DC motorun dönüş yönü de terslenmiş olur.

Bir motorun hızı rpm (rotations per minute - bir dakikada tamamlanan devir sayısı) ile ölçülür. Motorun hızı voltaja ve yüke bağlıdır. Bir DC motorun hızının voltaja ve yüke göre değişimini değerlendirmek için iki durum düşünülebilir. Bunlardan ilki; DC motora yük binmeyen ya da sabit bir yükün olduğu bir sistemdir. Böyle bir sistemde DC motorun hızı uygulanan voltaja bağlıdır ve voltaj arttıkça hız da artar. İkinci durum ise; DC motora binen yükün zamana ya da gerçekleştirilen göreve göre değiştiği bir sistemdir. Bu durumda DC motorun hızı yüke bağlı olacaktır. Yük arttıkça uygulanan güç de artar ve güç arttıkça hız azalır.

Küçük DC motorlar 1,5 V ile 48 V arasında değişen voltaj değerlerine sahip olarak bulunabilirler. Her bir DC motor için belirtilen voltaj değeri, o DC motorun kendi verilen hız, güç ve akım değerlerinde stabil çalıştığı voltaj değeridir. Robotlarda ve diğer sistemlerde DC motorları kullanırken de bu voltaj değeri, DC motora

verilecek maksimum çalışma voltajını belirlediği için önemlidir.

Güç bir motorun akımı ve voltajının çarpım değeridir. Ancak robot projelerinde ve mekanik sistemlerde bir motorun ürettiği kuvvetin tork (motorun dönme momenti) cinsinden değerlendirilmesi normaldir. Tork motorun dönme momentidir. Torku yüksek olan motor düşük olana göre daha güçlüdür. Tork motorun elektriksel ve mekanik karakteristiklerine ve motor şaftının yarıçapına bağlıdır. Bir motorun torku motora bağlanan dişli kutularıyla (redüktör) değiştirilebilir. Dişli kutuları hızın azaltılmasını ve gücün arttırılmasını sağlar. Örneğin; motor şaftının yarıçapının 10 katı yarıçapa sahip bir dişli motora eklendiğinde, motorun hızı 10 kat düşer ve gücü de 10 kat artar.

Robotikte, çeşitli boyutlarda ve redüksiyon oranlarında dişli kutuları motorun karakteristik özelliklerini istenilen işi yapabilecek düzeye getirmek için sıklıkla kullanılır. Bir motoru kullanırken torkunu bilmek önemlidir. Tork ve redüksiyon oranı bilindiğinde sistemin son çıkış gücü kolaylıkla belirlenebilir.

4. KOD YAZMAK NEDİR?

Bu konu hakkında aslında söylenebilecek çok fazla şey var. Kod yazmak kişilerin zihinlerinde oluşturdukları herhangi bir şeyi bilgisayar ortamında gerçekleştiğini görebilmelerini sağlamanın en basit yoludur. Bugün kod yazarak yani yazılım programlarıyla hem endüstri sektöründe hem eğlence, oyun sektörlerinde ve hatta akla gelmeyen bir çok sektörde daha istenilen her şeyi yapabiliyoruz.

Steve Jobs'un söylediği "Ülkedeki herkesin bilgisayar programlamayı öğrenmesi gerekiyor. Çünkü bu insana düşünmeyi öğretiyor" sözü gerçekten üzerinde düşünülmesi ve uğraşılması gereken bir konuya parmak basıyor. Bunu her ne kadar kendi ülkesi için söylemiş olsa da, kendi ülkemiz içinde bunun böyle olması gerektiği kanaatindeyim. İleride hangi meslekle uğraşıyor olursak olalım, artık bilgisayarlar her an her yerde önümüze çıkıyor. Bilgisayar programlayıcılığı ise yaptığımız işlerdeki iş yükümüzü inanılmaz derecede hafifletiyor ve vakitten tasarruf etmemize imkan sağlıyor. İşte sırf bu yüzden bile, herkesin, bir parça da olsa bilgisayar programlamayı bilmesi elzemdir.

Günümüzde kod yazmak için kullanılan onlarca dil var ve bu diller, aslında gerçek manada birer 'dil' gibidirler. İnsanlar nasıl Türkçe, İngilizce, Arapça gibi dilleri öğrenmeye can atıyorsa, aslında bu dilleri öğrenmeye de bir o kadar istekli olmalılar. Çünkü esasen bu dillerin çoğu birbirine benziyorlar ve bazı gramatik farklılıklar haricinde, neredeyse hepsinde, aynı mantık hakim. İşte bu yüzden Pseudo Coding tüm programlama dillerinin temeli denebilir. Çünkü pseudo code sayesinde, günlük yaşantımızda kullandığımız dil ile neyi anlatmak

istiyorsak, anlatabiliriz. Aslında mantık çok basittir. Bilinmesi gereken bir kaç tane yapı vardır. Bu yapıların kısaltmaları İngilizceden geçmiştir. Örneğin "**if-else**" yapısı. "**If**" İngilizcedeki kelime manasıyla Türkçede "**eğer**" demektir. Yani bir koşul bildirir. Bunu bir örnekle açıklayalım. Örnek cümlemiz; birey, eğer yaşı 18 in üstündeyse Türkiyede ehliyet alabilir. Bu Türkçe ifade pseudo coding sayesinde şu hale dönüşür:

```
if (yas > 18)
(ehliyet alabilirsin)
```

Bir önceki paragrafta bahsettiğim bir diğer yapı ise "else" di. Bu kelimede İngilizce kökenlidir ve manası Türkçede "aksi koşulda" olarak çevrilebilir. Bunu da yine üstteki aynı örnek cümle üzerinden pseudo code ile ifade edelim:

```
if (yas > 18)
(ehliyet alabilirsin)
                    else
(ehliyet alamazsın)
```

Kod yazarken sıkça kullanılan bir diğer yapı ise döngülerdir. Döngü-lerin mantığı ise bilgisayara bir işlemi istediğiniz sıklıkta yaptırmaktır. Bunu kod yazarak yaptırabilmek için "**while**" veya "**for**" anahtar kelimeleri kullanılır. "While" kelime manası gereği "belirlenen koşul devam ettikçe" demektir. "**for**" ise bildiğimiz gibi "**için**" demektir ve yine "**belirlenen koşullar için**" demektir. Örneğin, bilgisayarın art arda 10 defa bir oyunu

açmasını istiyorsu-nuz. Bunun için pseudo kodunuz şu şekilde olmalıdır:

```
kacdefa = 10
while (kacdefa > 0)
open (oyun)
                    kacdefa – 1
```

Pseudo kodumuzu kısaca açıklamak gerekirse, "**kacdefa**" diye bir değişken oluşturmuş olduk ve ilk baştaki değerini "10" yaptık. "**While**" döngüsü içerisindeki kontrolümüz ise bu "**kacdefa**" isimli değişkenin "0"dan büyük ol-masını inceliyor. Eğer "**kacdefa**" isimli değişkenimiz sıfırdan büyük olursa "**open**" anahtar kelimesiyle oyunumuz bilgisayara açtırmış oluyoruz. Ancak şöyle bir sorunumuz var. Eğer "**kacdefa**" isimli değişkenin tuttuğu değer döngünü içine her gittikten sonra 1 azaltmazsak döngümüz sonsuz olur ve eğer uzun süre farketmezsek bilgisayarımızın RAM ini aşırı yüklenme sonucu yakabiliriz. Bu en çok dikkat edilmesi gereken mevzulardan biridir. Bu 1, bir, azaltma işlemini de bildiğimiz ve matematikte kullandığımız eksi "-" işaretiyle yapıyoruz. Böylece "**kacdefa**" isimli değişkenimiz her döngüye girdikten ve oyun açıldıktan sonra 1 azalmış oluyor ve değeri 0 a ulaştığında döngümüzün koşulu artık sağlanmadığı için oyun bir daha açılmıyor.

"**for**" anahtar kelimesi de temelde bir döngü yapısıdır. Nasıl ki "**while**" yapısında belirlenen koşul sağlandığı sürece kodun iç kısmında bulunan komutun yapılması isteniyorsa, "**for**" yapısında da içinde belirtilen durum geçerli olduğu süre boyunca kodun kapsadığı komutun yapılması gerektiği emri veriliyor. "**while**" yapısı ile farkı ise üstteki örnekte bulunan ve bir nevi sayaç olarak

kullandığımız “**kacdefa**” isimli değişken, “**for**” yapısının dışında değil, içinde tanımlanıyor. Bunu bir önceki bilgisayara oyun açtırma örneğindeki gibi pseudo code şeklinde şöyle ifade edebiliriz:

```
for (kacdefa 10; kacdefa > 0; kacdefa-1)
                    run (oyun)
```

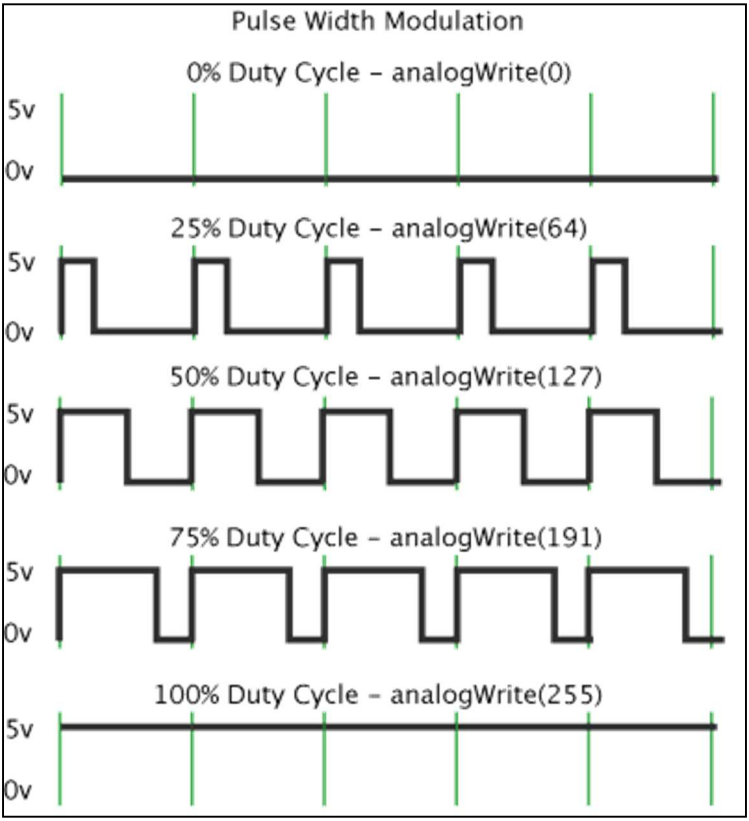
5. PWM (PULSE WİDTH MODULATİON - DARBE GENİŞLİKLİ MODULASYON) NEDİR?

PWM'in telekomünikasyon, güç kontrolü, ses amplifikatörleri, motor sürümü gibi birçok uygulama alanı vardır. Elektronik kontrol devrelerinde sıklıkla kullanılan bir yöntemdir. Analog sinyal ile digital sinyaller arasındaki fark, analog sinyalin genliği, frekansı değişebilirken, digital sinyal değeri 0-1 arasında değişebilir. Sinyalin değeri 1 olduğunda 5V çıkış alınır, 0 olduğunda 0V çıkış alınabilir. Örneğin; 2.43 V'luk bir çıkış gerilmini, arduino kartın çıkış pinini HIGH ya da LOW yaparak elde edemezsiniz. HIGH yapıldığında çıkış pininden 5V, LOW yapıldığında 0 V gerilim alınır. PWM ile digital çıkış sinyalinin duty cycle denen görev süresinin değeri ayarlanarak, ortalama gerilim elde edilir. Bu sayede Digital Analog dönüştürücü (DAC) gibi çalışması sağlanır.

PWM açılış ve kapanış süresi çok kısa olan bir anahtar olarak düşünülebilir. Anahtarın açık kaldığı süre boyunca yüke elektrik akımı gönderilirken, anahtar kapandığında yükün enerjisi kesilir. PWM sinyalinin frekansı Arduino kartlarda çoğunlukla 490 Hz dir. Resim 5.1 de PWM ile darbe genlik süresinin değiştirilmesi görülmektedir. Arduinodaki analogWrite() metoduna parametre olarak 0 değeri verildiğinde PWM çıkışlarından herhangi bir sinyal alınmaz. Görev süresi 0'dır. PWM değerleri 8 bitlik sayılarla ifade edildiğinden en fazla PWM değeri olan %100'ü analogWrite() metoduna 255 değeri verilerek elde edilir.

Arduino Uno kartı için 3,5,6,9,10 ve 11 nolu pinlerden PWM çıkışı alınabilmektedir. Arduino kartları üzerinde PWM pinleri ~ ile gösterilmiştir. analogWrite()

fonksiyonu iki parametre alır. İlk parametre PWM çıkış pini, ikinci parametre ise PWM Duty Cycle görev süresini belirten değerdir.



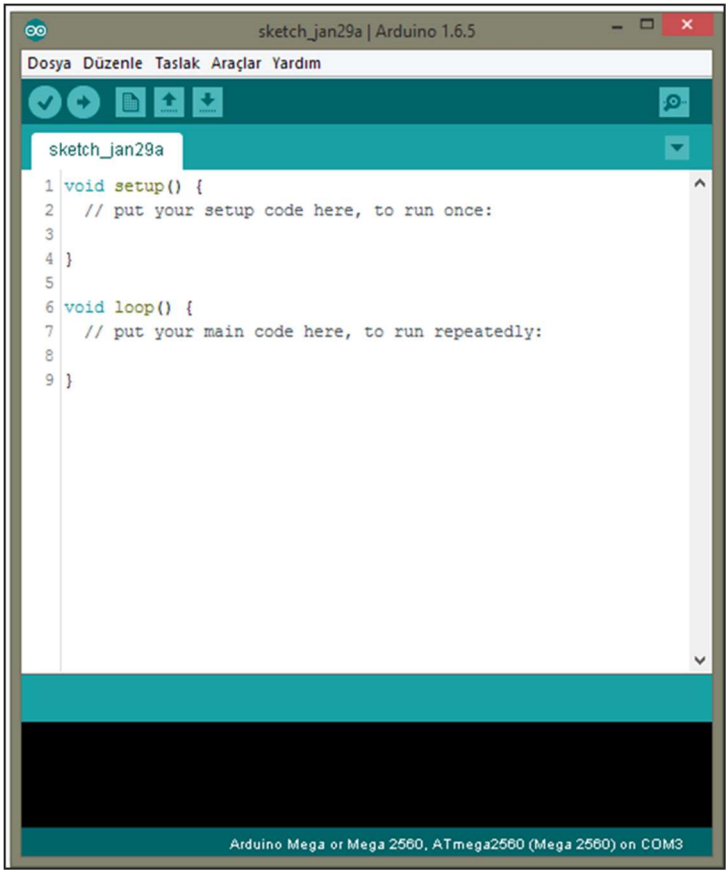
Çevre cihazları kontrol etmek için sıklıkla kullanılan PWM tekniği, analog sinyal gerektiren hoparlörden ses elde etmek gibi uygulamalarda yetersiz kalır. PWM tekniğinde Arduinoda 490 Hz / 980 Hz frekanslarında kare dalga üretilir. Üretilen kare dalga sinyalin duty cycle denen görev süresi yani 5V çıkış alınan süre,

saniye 490 kez tekrar eder. Bu yüzden ledlere gönderilen kare dalga sinyal değeri, gözün algılama frekansı olan 60'dan fazla olduğu için, gözler ledin açılıp kapandığını algılayamaz.

6. ARDUİNO DERLEYİCİSİ

6.1 Arduino Derleyicisi Kurulumu

Öncelikle Arduionun resmi sitesinden, programalama için kullanacağımız derleyicisini indirip kuralım. https://www.arduino.cc/en/main/software# adresinden gerekli indirmeyi ücretsiz olarak yapabilirsiniz



Programı kurup açtığımızda karşımıza üsteki gibi bir ekran çıkacaktır. Gördüğünüz gibi Arduino derleyicisi sade bir arayüze sahip. İçerisinde hazır olarak sunulmuş 2 method kalıbı bunlar bulunmaktadır.
setup methodu içerinde yazdıklarımız, kodumuzu derleyip yükledikten sonra 1 kereye mahsus olarak çalışmaktadır ve burada yalnızca tanımlaması yapılacak değişkenler yazılır. loop methodu altında ise asıl çalışılmak istenen kod kısmı eklenir. Buraya yazılanlar baştan sona sürekli olarak tekrar edilir.
Araçlar kısmından hangi Arduino türünü kullanıyorsak onu önceden belirtmemiz gerekiyor. Ayrıca kodu yüklerken kullandığımız yani Arduinoyu bilgisayarımıza bağladığımız USB port ismini (COM5, COM6.. gibi) de önceden ayarlamamız gerek. (bkz. Araçlar>Kart ve Araçlar>Port)

1 numara ile gösterilen ‘tik’ işareti kodumuzu yüklemeden önce derleyip yazım hatası olup olmadığını kontrol etmemize olanak sağlıyor. Böylece hatamızı ilk elden görüp düzeltebiliyoruz.
2 numara ile gösterilen ‘ok’ işareti de kodumuzdan emin olduktan sonra Arduino muzu programlamamızı sağlıyor.
3 numara ile gösterilen ‘büyüteç’ işareti ise Serial Port a erişim sağlamamıza yarıyor.
Arduino IDE bize aynı zamanda Dosya>Örnekler tabı altında hem kendi hem de eklenen kütüphanelerin örnek kodlarına erişimimize imkan sağlıyor.

6.2 Arduino Derleyicisine Kütüphane Ekleme

Deneylerimizde bazen harici cihazlar kullanmak durumunda kalıyoruz. Servo Motor, Sıcaklık sensörleri, Mesafe sensörleri gibi bu harici cihazları, Arduinodan verdiğimiz komutlarla yönetmek istiyorsak bazı kod bloklarına ihtiyacımız olur. Bunlar Arduino kütüphanesi olarak tanımlanırlar ve işimizi çok kolaylaştırırlar. Mühim olan kullanılacak olan cihaza uygun kütüphanenin bulunmasıdır. Bunu yapabilmek için kullandığımız cihazın bilgilerini örneğin "DS18B20 Sıcaklık Sensörü Arduino Kütüphanesi" şeklinde internette arattığımızda kütüphanesini indirebilirsiniz. İndirilen dosya rar veya zip arşivi ise yalnızca içindeki ana klasörü "C:\Program Files (x86)\Arduino\libraries" uzantısına taşıyınız. Bu sayede kütüphaneniz artık Arduino derleyiciniz tarafından tanınmış olacaktır.
Arduino derleyicisinde bir kütüphaneyi çağırmak için ise "#include kütüphaneismi" kodun üst kısmına eklenmelidir.

7. ÖRNEK DENEYLER

7.1 Deney #1 Led Yakma

Bu deneyimizde, daha önce Led kısmında da belirttiğim gibi ledlerin deneylerde kontrol amaçlı nasıl kullanılabileceğini anlamak için, belki de en basit Arduino uygulamalarından biri olan 'Blink' uygulamasını yapacağız.

Deney Malzeme Listesi:

- 1 adet Led
- 1 adet 300 Ω direnç
- Arduino UNO
- Jumper
- 9V pil
- Breadboard
- Digital Multimetre

İşe, öncelikle devre elemanlarımızı breadboard üzerinde, Resim 7.1'deki gibi yerleştirerek başlayalım. Gördüğünüz gibi led ile birlikte bir de direnç kullandık. Devrede kullandığımız bu direnç, Arduino'dan gelen akımın bir kısmını kendi üzerinde harcayacaktır. Bu sayede ledimiz daha az akım çekip korunmuş olacak. Aksi halde, bazı durumlarda lede fazla akım gittiği için ledin zarar görmesi hatta patlaması söz konusu olabilmekte ve bu malzemenin içindeki kimyasallar uzun vadede de olsa insan sağlığına zararlı olabilir. Bu yüzden düşük değerli de olsa, devrelerimizde kullandığımız ledlerin bir bacağına direnç bağlanması isabetli olacaktır.

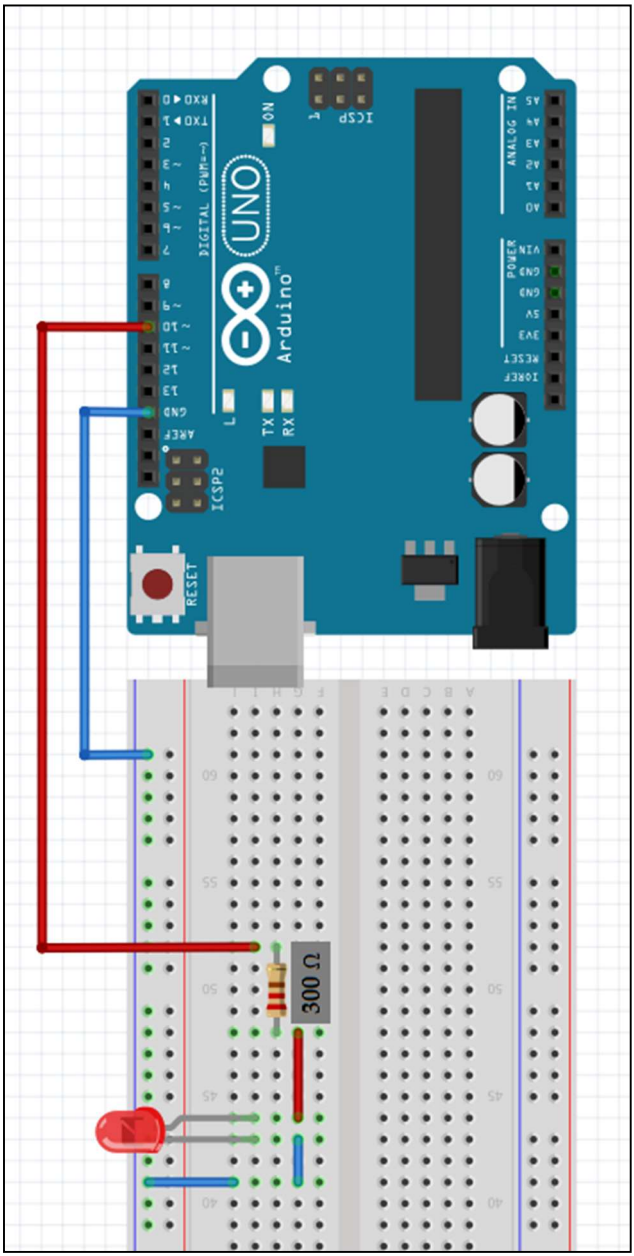
Bunun dışında, 9V pile yalnızca, Arduino'yu bilgisayar bağlantısı olmadan kullanacaksak ihtiyaç duyacağız. Çünkü, Arduino fazla güç istemeyen devre tasarımlarımıza yetecek akımını bilgisayardan çekip hem kendine hem devreye güç sağlar (Dijital Pinlerden ortalama 40 mA).

Kod 7.1: Led Yakma, Blink

```
int led=10; // ledimizin sinyal alacağı
//arduino digital
//pin numarası
void setup() {

  pinMode(led,OUTPUT);
}

void loop() { //döngünün olacağı kod alanı

  digitalWrite(led,HIGH);// led e yan komutu
variliyor
  delay(500);//led 500 ms boyunca yanık kalsın
komutu
  digitalWrite(led,LOW);//led e sön komutu
veriliyor
  delay(500);//led 500 ms boyunca sönük kalsın
komutu
}
```

Yazdığımız kodda öncelikli olarak, ledimizin Arduino üzerinden hangi dijital pine bağlayacağımızı belirttik. Ardından, led değişkeniyle tuttuğu-muz pin numarasını kullanarak setup fonksiyonununda ledimizi OUTPUT yani çıkış olarak tanımladık. loop fonksiyonunda ise ledimizi yakmak için gerekli dijital sinyali gönderen digitalWrite komutunun içine HIGH diyerek 5V'luk

pulse gönderdik. Böylece ilk olarak ledimizi yaktık. delay ise hatırlayacağınız üzere kodu, içerisinde yazan sayı kadar, milisaniye olarak, programı bekletiyordu. Biz kodumuzda içine 500 yazarak, yarım saniye kadar programın beklemesini yani ledin yanık durumda kalmasını ayarladık. Diğer satırda ise ledimize LOW yani 0V'luk pulse göndererek sönmesini sağladık. Ve yine delay kullanarak programı beklettik ve led yarım saniye kadar sönük kaldı. Loop fonksiyonu içindeki işlem, siz Arduino'ya yeni bir kod yükleyene veya enerjiyi kesene kadar devam edecektir.


UNO
Arduino
300 Ω
RESET
ICSP2
ICSP
ON
L
TX
RX
DIGITAL (PWM~)
ANALOG IN
POWER
AREF
GND
RESET
3V3
5V
VIN

7.2 Deney #2 Buton ile Kontrol

Bu deneyimizde bir buton yardımıyla led yakacağız. Devremizde 2 adet led kullanacağız ve bu ledlerden birisi normal koşulda yanarken, butona basıldığında sönüp diğer ledimiz yanacak. Butona basılı tutulduğu sürece 2. led yanık durumda olacak ve elimizi butondan çekince, ledlerimiz, ilk duruma dönecek. Şimdi gelin bunu çalışır vaziyette iken inceleyelim.

Deney Malzeme Listesi:

- 2 adet Led
- 1 adet buton
- 2 adet 300 Ω direnç
- 1 adet 4.7k Ω direnç
- Arduino UNO
- Jumper
- 9V pil
- Breadboard
- Digital Multimetre

Elimizdeki deney malzemelerini Resim 7.2'ye uygun biçimde breadboard üzerine yerleştirin. Ledlerimizin önüne neden direnç bağladığını ilk deneyimizde öğrenmiştik. Bu sefer bir direnç de butonun toprağa giden bacağına yerleştirilmeli. Bunun sebebi, butonların çok güvenilir bir devre ele-manı olmamasından kaynaklanıyor. Butona basıldığında normalde istenilen Arduinodan gelen bütün akımın buton üzerinden geçmesidir. Ancak, butonlar mekanik yapıları gereği bu

işi anlık olarak yapamazlar ve Arduino gibi mikroişlemci taşıyan cihazlar butonun gönderdiği verilerdeki değişimleri ve küçük zamanlı salınımları algılayabilir durumdadırlar. Bu durum devremizin verimini düşürür. Dijital sinyalleri hatırlarsanız, yalnızca 0V ve 5V değerlerini algılayıp bunlara göre 0 ve 1 değerleri döndürürler. Butonların sebep oldukların küçük zamanlı elektrik salınımları da veri kaybına sebep olur. Bunu engellemek için de butona Arduinodan gelen voltaj pinine veya toprağa giden bacağına bir direnç bağlanır. Direnç, toprağa giden bacağa bağlandıysa 'pull-down', öteki bacağa bağlandıysa 'pull-up' direnci olarak isimlendirilir. Bu direncin değeri genelde 4.7kΩ olarak tercih edilir. Tam değerini hesaplamak için hepimizin bildiği Ohm Yasası kullanılır. Ohm Yasası V = I x R olup V dijital 1 manasında 5V olarak alınır ve akım 'I' kaç amperlik akım geçeceğini belirtir ki, devremizde 1 mA'e yakın bir değer olarak alabiliriz. Bu işlemin sonucunda direnç değerimiz 5kΩ olmalıdır. Ama piyasada 4.7kΩ dirençler daha çok bulunuyor bu yüzden onu tercih ediyoruz. Bu küçük miktardaki farkı, devre tarafından amorti edilebilir düzeydedir.

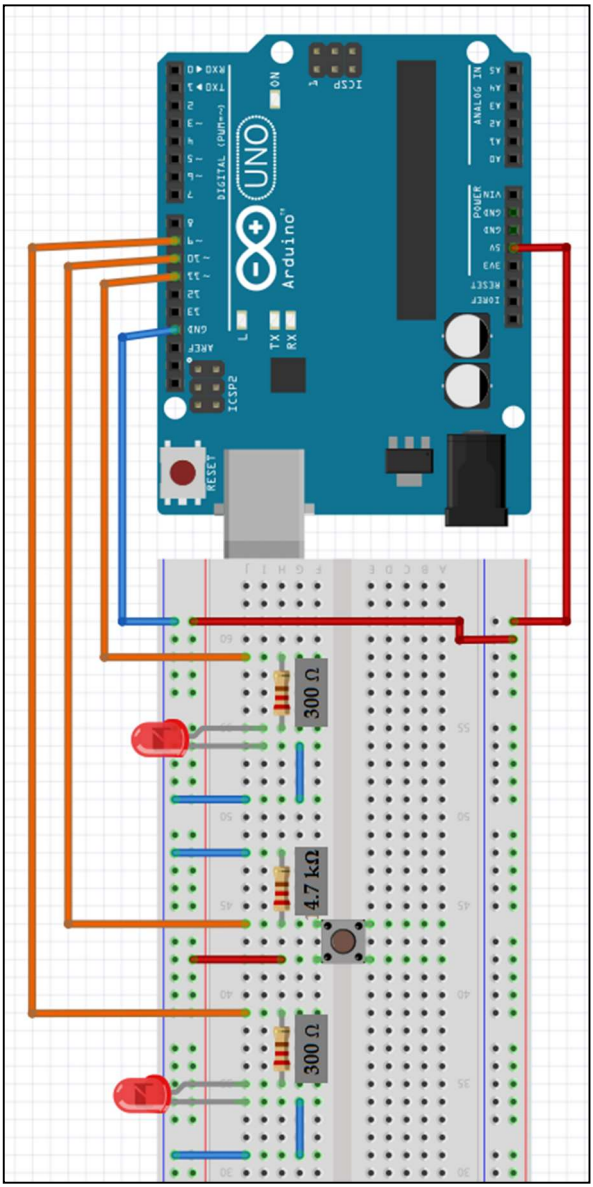
Kod 7.2: Buton ile Kontrol

```
int buton=10;
int led1=9;
int led2=11;
void setup() {

  pinMode(led1,OUTPUT); //led pinimizi çıkış ayarladık
  pinMode(led2,OUTPUT); //led pinimizi çıkış ayarladık
  pinMode(buton, INPUT); // buton pinimizi giriş
ayarladık

}
void loop() {
  butonDurum=digitalRead(butonPin);//butondan değer
okuma
  if(butonDurum==HIGH){
    digitalWrite(led1,HIGH);//birinci ledimiz yanıyor
    digitalWrite(led2,LOW);//ikinci ledimiz sönüyor
  }
  if(butonDurum==LOW){
    digitalWrite(led1,LOW);//birinci ledimiz sönüyor
    digitalWrite(led2,HIGH);//ikinci ledimiz yanıyor
  }
}
```

Devremizi breadboard üzerinde kurduktan sonra kodumuzu yazmaya başlayabiliriz. Öncelikle, her zaman yapacağımız gibi, Arduino kodumuzda kullanacağımız değişkenlerimizi tanımlayarak işe başlayalım. Devremizde Arduino'nun 3 adet dijital pinini kullandık. Bunların ikisi ledler, diğeri de bu-ton içindi. buton, led1

ve led2 değişken isimleriyle bu dijital pinlerin numaralarını kod içerisinde tanımlamış olduk. setup fonksiyonu içerisinde de bu değişken isimlerimizi kullanarak, led1 ve led2 'yi OUTPUT, butonu ise INPUT olarak ayarladık. Böylece Arduino, 9 ve 11 nolu pinlerden veri gönderirken, 10 nolu pinden veri okuması yapacak. loop fonksiyonu içerisinde de butonDurum şeklinde bir başka değişken oluşturduk. Bu değişkenimiz, digitalRead 10 numaralı pinden gelen dijital veriyi tutuyor. Bu aşamadan sonra ise, 'if' yapısıyla kontrol kısmı geliyor. Her iki 'if' yapısında da butonDurum değişkeninin du-rumu inceleniyor. Eğer butona basılırsa Arduino'ya 5V yani HIGH komutu gidecektir. Bu sayede birinci ledimiz yanarken ikinci ledimiz sönecek. Diğer durumda ise Arduino'ya 0V yani LOW komutu gidecek ve ilk led sönüp ikincisi yanmaya başlayacak.

Dipnot: Eğer Pull-up veya Pull-down direnci koyamayacak olursak, bunu kodda pinMode (buton, INPUT_PULLUP) şeklinde belirterek halledebiliriz.


UNO
Arduino
300 Ω
4.7 kΩ
300 Ω

7.3 Deney #3 Ledli Karaşimşek Devresi

Bu deneyimizde, daha önce açıkladığımız 'for' döngüsünü örnekleme şansı bulacağız. Ayrıca ledlerimiz tanımlarken dizilerden faydalanacağız. Diziler birçok değişken tipinde olabilmektedir. Örneğin, C dilinde bir kelime tanımlamak istiyorsak başvurabileceğimiz yöntemlerden biri de karakter dizisi oluşturmaktır.

```
char kelime[] = {'A','r','d','u','i','n','o'};
```

Deney kodumuzda ise integer değişken tipinde bir dizi oluşturacağız ve böylece devrede kullanacağımız çok sayıdaki ledimizin dijital pin numaralarını tek tek tanımlamak zorunda kalmayacağız.

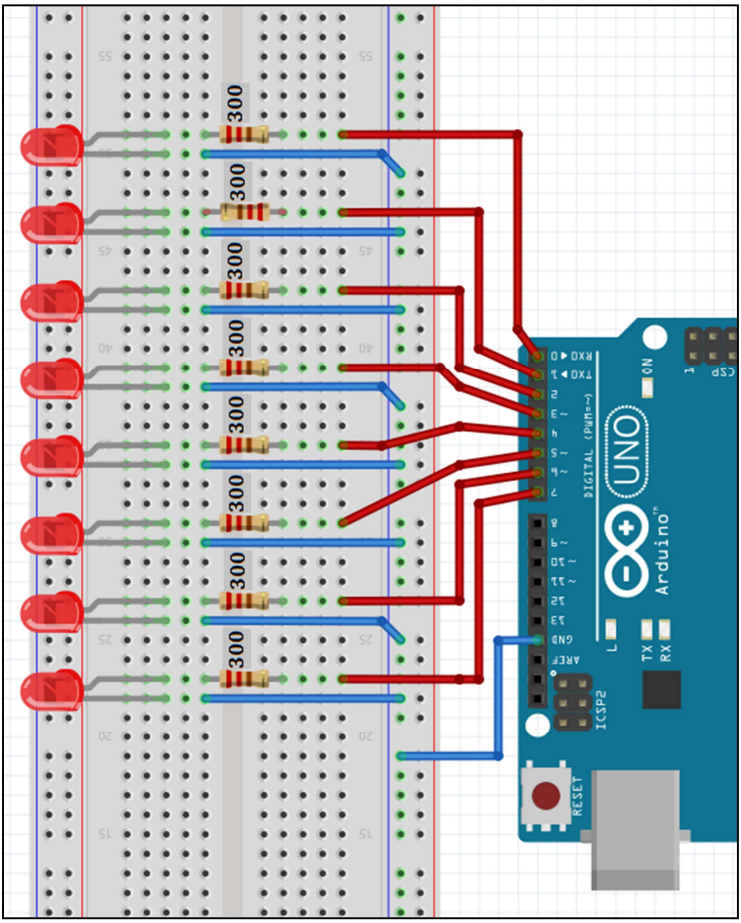
Deney Malzeme Listesi:

- 8 adet Led
- 8 adet 300 Ω direnç
- Arduino UNO
- Jumper
- Breadboard

Kod 7.3: Ledli Karaşimşek

```
int ledPin[] = {0, 1, 2, 3, 4, 5, 6, 7};
void setup() {
  for (int i = 0 ; i < 8 ; i++) {
    pinMode(ledPin[i], OUTPUT);
  }
}
void loop() {
  for (int i = 0 ; i < 8 ; i++) {
    digitalWrite(ledPin[i], HIGH);
    delay(150);//milisaniye bekleme
    digitalWrite(ledPin[i], LOW);
  }
  for (int i = 7 ; i >= 0 ; i--) {
    digitalWrite(ledPin[i], HIGH);
    delay(150);
    digitalWrite(ledPin[i], LOW);
  }
}
```

setup() içerisindeki for döngüsü sayesinde ledlerimizi pinMode fonksiyonu kullanarak toplu olarak tanımlamış olduk. for döngüsü içerisinde i değişkenini oluşturduk ve süslü parantezlerin arasındaki işlemi, içerisindeki eşitlik sağlanana kadar yapar. Böylece ilk döngümüzde ledPin[0] bizim ilk ledimizi, ledPin[7] ise son ledimizi tanımlamış oluyor. loop() fonksiyonu da bildiğiniz üzere, Arduinodan elektriği kesene kadar çalışmaya devam bir fonksiyon olarak tanımlanmıştı. İçersindeki iki for döngüsü sayesinde ilk ledimizden son ledimize doğru, önce bir yönde sonra diğer yönde sürekli olarak yanıp sönen ledlerimizi oluşturduk.


300
300
300
300
300
300
300
300
UNO
Arduino
DIGITAL (PWM~)
ON
L
TX
RX
RESET
ICSP2
ICSP

7.4 Deney #4 Serial Port Kullanma

Bu deneyimizde, elektronikte önemli bir yer tutan cihazlar arası haberleşmeyi kullanarak Arduino'ya komut vereceğiz ve bu komutlarla da devremizdeki ledlerin istediğimiz sırada yanmasını sağlayacağız. C programlama ile Arduno başlığı altında da incelediğimiz Serial Port özelliğini, 'if' yapısıyla harmanlayacağız.

Deney Malzeme Listesi:

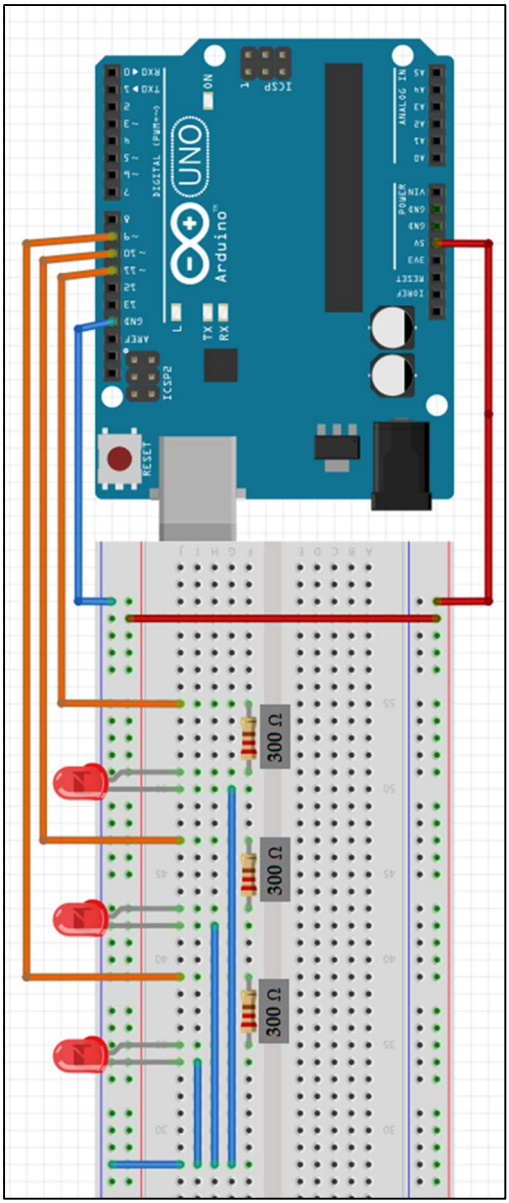
- 3 adet Led
- 3 adet 300 Ω direnç
- Arduino UNO
- Jumper
- 9V pil
- Breadboard
- Digital Multimetre

Kodumuzu yazmaya her zaman olduğu gibi, gerekli pin tanımlamalarımızla başlıyoruz. setup fonksiyonu içinde de yine ledlerimizi bağladığı-mız pinlerin mod ayarını yapaıyoruz. **Serial**.begin(9600) komutuyla Arduino'nun serial portundan veri alış-verişi sağlamak için gerekli başlatma ayarını yapıyoruz. Bu komutun içinde yazan değer, Arduino'nun Serial Port aracılığıyla alabileceği saniye başına bit değerini gösterir. Biz yaptığımız çalış-malarda, genelde, 9600 değerini tercih ediyoruz. loop fonksiyonuna geldiği-mizde ise **Serial**.read() komutu karşımıza çıkıyor.

Kod 7.4: Serial Port Kullanma

```
int led1=9;
int led2=10;
int led3=11;
char hangiLed;//degişkenlerimizi tanımladık
void setup() {
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);//ledlerimize mod ayarı yaptık
  Serial.begin(9600);//iletişimin başlangıcı için gerekli
                          //saniyede alabileceği bit veri sayısı*
}
void loop() {
  hangiLed=Serial.read();//Portta girdiğimiz veriler
//okunuyor
  if(hangiLed == 'a'){//porttan alınan veriyle
                //karşılaştırma
    digitalWrite(led1,HIGH);
  }
  if(hangiLed == 'b'){
    digitalWrite(led2,HIGH);
  }
  if(hangiLed == 'c'){
    digitalWrite(led3,HIGH);
  }
  if(hangiLed == 's'){
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
    digitalWrite(led3,LOW);
      }
  else
     Serial.write("\n!! GECERSIZ GIRIS !!\n");
 }
```

Bu komut Serial Porttan alı-nan verinin ‘byte’ olarak okunmasını sağlıyor. Ve biz kodumuzda aldığımız bu veriyi ‘char’ tipinde hangiLed isimli değişkende tutuyoruz. Tuttuğumuz bu değerle de ‘if’ yapıları içerisinde her bir led için karşılaştırma yapıyoruz. Programımız kullanıcını yalnızca ‘a’, ‘b’, ‘c’, ‘s’ değerlerini girdiği durumda çalışmaya ayarlandı. İlk üç harften birisi girildiğinde, her biri için ‘if’ yapısının içinde tanımladığımız led yanmaya başlayacak. Eğer söndürmek istersek de, Serial Portumuzdan ‘s’ verisini göndermemiz yeterli olacak. Programımızda eğer bu dört karakterden başka bir veri girildiğinde, Porta **Serial**.write fonksiyonuyla birlikte uyarı yazısı yazdırılıyor.


UNO
Arduino
300 Ω
300 Ω
300 Ω

7.5 Deney #5 POT Kullanma

Bu deneyimizde POT Nedir başlığı altında incelenmiş olan Potansiyo-metreden Analog Veri alınarak Ledin parlaklığının nasıl değiştiğini incelemiş olacağız. POT, Analog Input elemanlarından birisidir ve üzerinden yapılan her bir değişim, Arduino tarafında 0-1023 arası bir değerle derecelendirilir. Ancak Ledlerimizi Analog Output elemanı olarak kullandığımızda çalışma aralıkları 0-255 arası değerlendirilir. Bu yüzden, kodumuzda bununla ilgili bir ayarlama yapmamız gerekecek. Öncelikle malzemelerimiz toparlayıp Devremizi Breadboard üzerinde, Resim 7.5'e göre hazırlayalım.

Deney Malzeme Listesi:

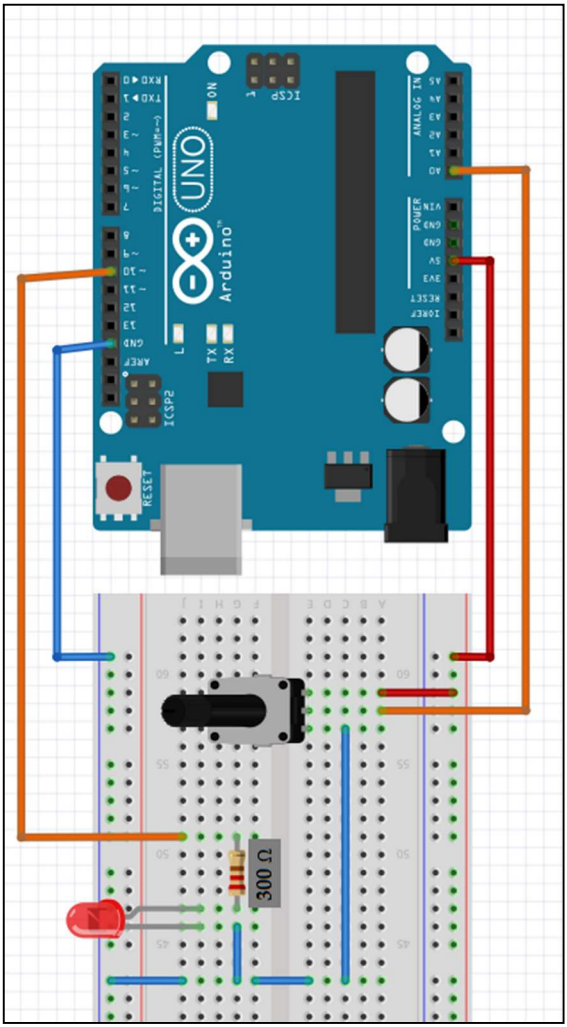
- 1 adet Led
- 1 adet 300 Ω direnç
- 1 adet Pot
- Arduino UNO
- Jumper
- 9V pil
- Breadboard
- Digital Multimetre

Kod 7.5: POT Kullanma

```
int led=10;
int pot=A0;
int potDegeri;//değişken tanımları

void setup() {
  pinMode(pot,INPUT);//kullanılan pinlerin mod ayarları
  pinMode(led,OUTPUT);
  //Serial.begin(9600);//denemek için yorumu kaldırın
}
void loop() {
   int potDeger=analogRead(pot);//analog veri okuma
  //Serial.write("\n"+ potDeger);//denemek için
  potDeger=map(potDeger, 0, 1023, 0, 255);//aralıkları
//düzenleme işlemi
  analogWrite(led, potDeger);//
  //delay(10);//denemek için yorumu kaldır
}
```

led ve pot isimli değişkenlerimizle Arduino üzerinde bağlantı yaptı-ğımız pinleri programda belirttik. setup fonksiyonu içerisinde pot umuzu INPUT, ledimizi de OUTPUT olacak şekilde ayarladık. loop fonksiyonunda ise A0 Analog pinine bağlı olan pot umuzdan aldığımız Analog Verileri potDeger isimli 'int' tipindeki değişkenimiz içinde 0-1023 arasında derecelendirip sakladık. Ancak Led'ler analog verileri 0-255 arasında yansıtabilir. Bu yüzden Arduino kütüphanesinde bulunan map() fonksiyonunu kullandık. Bu fonksiyonun içerisine ilk yazacağımız, değiştirilen değerin, hangi değişkenden alınacağıdır. Bundan sonra ise ilk başta değiştirilmek istenen aralık ile dönüştürmek istediğimiz aralık sırayla yazılır. Bu fonksiyon sayesinde pottan veri alıp kolayca

ledimizi yakabileceğiz. potDeger ‘in tuttuğu sayı değerini analogWrite ile ledimize gönderip pottaki değişime göre ledimizi yakacağız. Programda yorum olarak ‘//’ sonra yazılan kod parçalarını açarak, Serial Portta, Pot’un değerindeki değişimlerini gözlemleyebilirsiniz.


UNO
Arduino
300 Ω

7.6 Deney #6 LDR Kullanma

Bu deneyimizde LDR kullanarak, ortamın ışık miktarına bağlı olarak değişen bir aydınlatma sistemi kuracağız. Bu sayede ortam karanlıkken bile, ledler sayesinde belirli bir alanı ışıklandırmış olacağız. LDR'de POT gibi bir Analog Input elemanıdır. Bu yüzden çevreden aldığı ışık şiddeti verisini 0-1024 arası bir değerle Arduino da değerlendirilir.

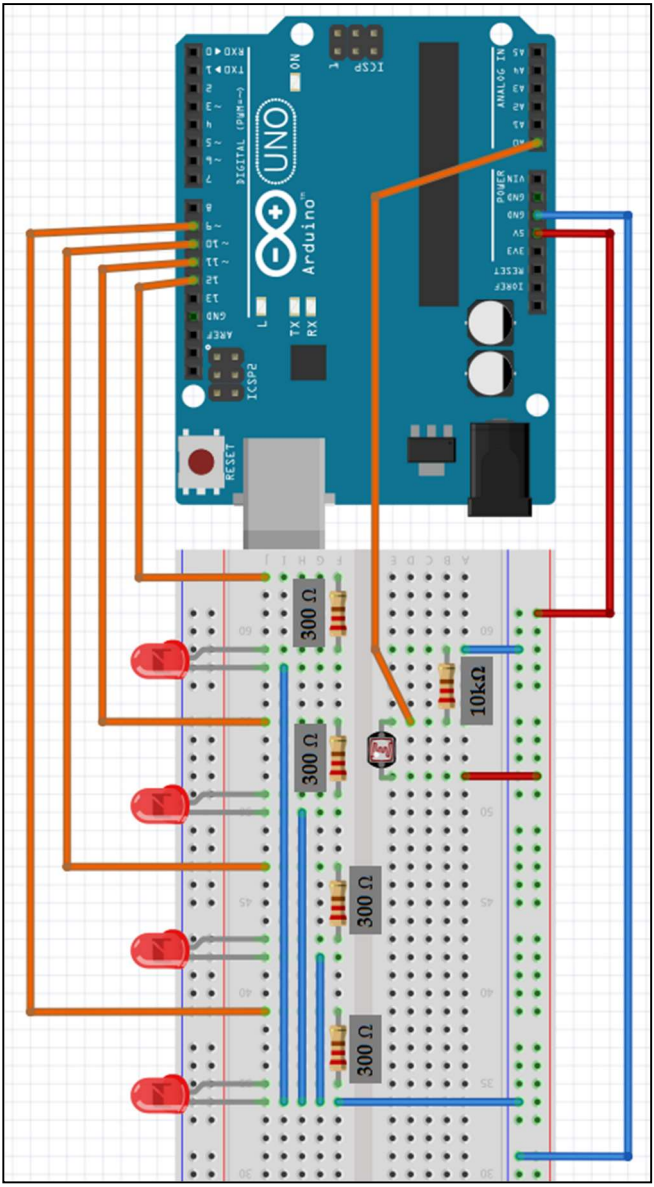
Deney Malzeme Listesi:

- 4 adet Led
- 4 adet 300 Ω direnç
- 1 adet 10k Ω direnç
- 1 adet LDR
- Arduino UNO
- Jumper
- 9V pil
- Breadboard
- Digital Multimetre

Kod 7.6: LDR Kullanma

```
int led1=9;
int led2=10;
int led3=11;
int led4=12;
int ldr=A0; //değişken tanımları
void setup() {
 pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
   pinMode(led3,OUTPUT);
    pinMode(led4,OUTPUT);
     pinMode(ldr,INPUT);  //giriş ve çıkış mod
ayarları
}
void loop() {
 int  isiksiddeti = analogRead(ldr);//ldrden alınan
veri
  if(isiksiddeti<256){//verilere göre aralık
        kıyası
    digitalWrite(led1,HIGH);
    digitalWrite(led2,LOW);
    digitalWrite(led3,LOW);
    digitalWrite(led4,LOW);
  }
  if(isiksiddeti<512 && isiksiddeti>=256){
    digitalWrite(led1,HIGH);
    digitalWrite(led2,HIGH);
    digitalWrite(led3,LOW);
    digitalWrite(led4,LOW);
  }
  if(isiksiddeti<768 && isiksiddeti>=512){
    digitalWrite(led1,HIGH);
    digitalWrite(led2,HIGH);
    digitalWrite(led3,HIGH);
    digitalWrite(led4,LOW);
  }
  if(isiksiddeti>=768){
    digitalWrite(led1,HIGH);
    digitalWrite(led2,HIGH);
    digitalWrite(led3,HIGH);
    digitalWrite(led4,HIGH);
  }
}
```

Değişken tanımlarımızla birlikte, tüm giriş ve çıkış elemanlarımızın mod ayarını yaptık. loop fonksiyonu içerisinde öncelikle isiksiddeti isminde ‘int’ türünde bir değişken oluşturduk ve LDR den gelen Analog verilerimizi bu değişken üzerinde tuttuk. Alt satırlarda ise 4 tane ‘if’ kullanarak LDR den alınan bu veriyi, sabit değerlerimizde karşılaştırma yaparak sınıflandırdık. Önceden de belirttiğim gibi, LDR Arduinoda 0-1023 arası bir değerle döndürülür. Kurduğumuz ‘if’ karşılaştırmalarındaki sabit değerlerimizi de bu aralığı 4 parçaya bölerek ayarladık. Ortamın ışık şiddetinin azalmasıyla birlikte yanan led sayısında artış olacak.


UNO
Arduino
300 Ω
300 Ω
300 Ω
300 Ω
10kΩ

7.7 Deney #7 Termometre Yapalım

Bu deneyimizde giriş olarak Analog veri değil de Dijital veri gönderen bir sıcaklık sensörü olan DS18B20 kullancağız. Bunu sıcaklık sensörünün mu-adili olan ve analog veri gönderen LM35'i de kullanabilirsiniz. İşlevler aynı ancak programı ve devresi elbetteki farklı olacaktır. DS18B20 Arduinodaki kü-tüphanesi sayesinde oldukça kullanışlıdır. Ancak bu kütüphaneyi internetten indirip Arduino ana kütüphanesinin içine koymanız gerekmektedir. Bununla ilgili bilgiyi 6. kısım altında bulabilirsiniz.

Deney Malzeme Listesi:

- 4 adet Led
- 4 adet 300 Ω direnç
- 1 adet 4.7k Ω direnç
- 1 adet DS18B20 Sıcaklık Sensörü
- Arduino UNO
- Jumper
- 9V pil
- Breadboard
- Digital Multimetre

Kod 7.7: Termometre Yapalım

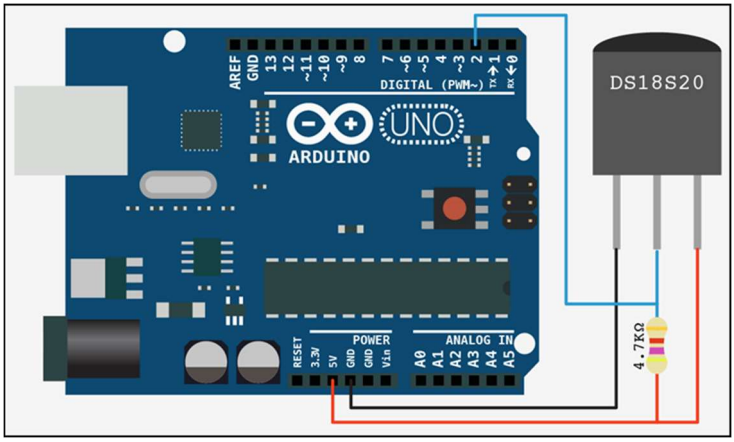
```
//kütüphaneleri ekleme komutu
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
int led1=9;
int led2=10;
int led3=11;
int led4=12;
void setup(void) {
 pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(led4,OUTPUT);
  //Serial.begin(9600);
  //Serial.println("Arduino Dijital Sicaklik ");
  sensors.begin();
}
void loop(void)
{
  sensors.requestTemperatures();
  /*Serial.print("Sicaklik: ");
Serial.println(sensors.getTempCByIndex(0)); */
   if(sensors.getTempCByIndex(0)<21.0){
    digitalWrite(led1,HIGH);
    digitalWrite(led2,LOW);
    digitalWrite(led3,LOW);
    digitalWrite(led4,LOW);
  }
   else if(sensors.getTempCByIndex(0)>=21.0 &&
sensors.getTempCByIndex(0)<24.0){
    digitalWrite(led1,HIGH);
    digitalWrite(led2,HIGH);
    digitalWrite(led3,LOW);
    digitalWrite(led4,LOW);
  }
```

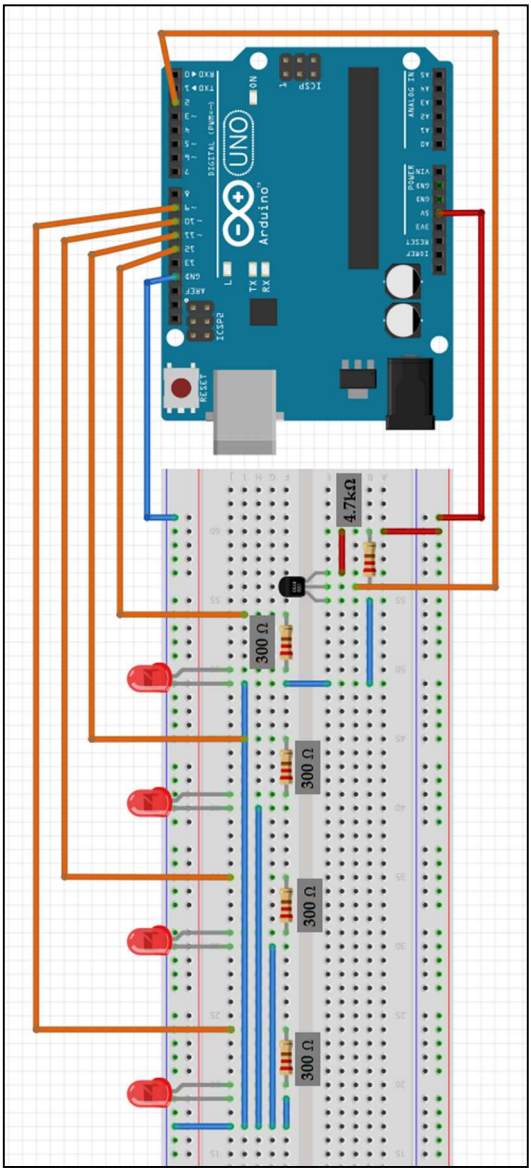
```
  else if(sensors.getTempCByIndex(0)>=24.0 &&
sensors.getTempCByIndex(0)<28.0){
   digitalWrite(led1,HIGH);
   digitalWrite(led2,HIGH);
   digitalWrite(led3,HIGH);
   digitalWrite(led4,LOW);
  }
   else if(sensors.getTempCByIndex(0)>28.0){
   digitalWrite(led1,HIGH);
   digitalWrite(led2,HIGH);
   digitalWrite(led3,HIGH);
   digitalWrite(led4,HIGH);
  }
delay(1000);//saniyede bir veri alması için

}
```

Programımızın başında eklememiz gereken 2 tane kütüphane bulunuyor. Bunlar DS18B20 isimli sıcaklık sensörümüzden alınan verilen kolay bir şekilde, anlaşılabilir olmasını sağlayan bir takım kodları içeriyor. Analog verileri Arduino ile 0-1023 arasında bir değerle örtüştürüp değerlendirebiliyorduk ancak bu sensörümüzün iç yapısı biraz daha karmaşık ve dijital veri sağlıyor. Kütüphane eklemeyi 5. konu başlığından bakıp kolayca yapabilirsiniz.

Ledlerimizi dijital pinlerimizde ve sensörümüzü de yine bir dijital pinde olacak şekilde tanımlıyoruz. setup fonksiyonunda ledlerimizin modlarını ayarladık. loop fonksiyonunun başında requestTemperatures(); fonksiyonuyla sensörümüzden veri alınması sağlanıyor. Alınan bu veri Santigrat cinsinden sıcaklık değeri olarak, yukarıda eklenen kütüphaneler tarafından çevriliyor ve ortamın sıcaklığı ölçülüyor. Daha sonrasında ise ortamın sıcaklı-ğına göre belirli sıcaklık değerleriyle

karşılaştırma yaparak, en yüksek sıcaklıkta en çok, en düşük sıcaklıkta ise en az sayıda led yanacak şekilde ‘if’ koşullu ifadelerimizi yazıyoruz. Kod bitiminde ise saniyede bir defa, sensörden veri alınmasını sağlamak için delay(1000) diyoruz ve programımız böylece hazır hale geliyor. Koddaki yorum kısımlarını kaldırıp sensörden alınan sıcaklık bilgilerini Serail Port üzerinden de takip edebilirsiniz.




UNO
Arduino
4.7kΩ
300 Ω
300 Ω
300 Ω
300 Ω

7.8 Deney #8 Ultrasonik Sensör ile Mesafe Algılama Devresi

Bu deneyimizde, insan kulağının algılayacabileceği en yüksek frekans seviyesi olan 20kHz'in üzerinde olan ultrasonik ses dalgalarını mesafe algılamak için kullanan bir sisteme sahip olan Ultrasonik sensör kullanacağız. Çalışma mekaniğinden kısaca bahsetmek gerekirse, Ultrasonik sensörlerde 40kHz'de çalışan hem alıcı, hem de verici sensör bulunmaktadır. Sensörün vericisinden gönderilen ses dalgası bir cisme çarptıktan sonra alıcısına geri yansır. Bu yansıma sayesinde gidip-gelme süresi ölçülerek uzaklık hesaplanır. Uzaklık ve yön çözünürlüğünün yüksek olmasını sağlayan ise elektromanyetik dalgaların hızları $3x10^8$ m/s iken sesin oda koşullarındaki hızının yaklaşık 344 m/s olmasıdır. Bu düşük hız sayesinde çözünürlük artar ve daha doğru ölçüm sonuçları elde edilir. Park sensörleri, araç alarm sensörleri, otomatik kapı sensörleri, robotlar ve navigasyon cihazları ultrasonik sensörlerin yaygın kullanım alanları olarak gösterilebilir.

Kod incelendiğinde pulseIn isimli bir fonksiyonun kullanıldığı görülecektir. Bu fonksiyon belirtilen digital pindeki HIGH veya LOW olma durumunun ne kadar sürdüğünü mikrosaniye cinsinden geri döndürür.

Ultrasonik sensörün özellikleri:

- Çalışma voltajı = 5V DC
- Çalışma akımı = 15mA
- Çalışma frekansı = 40Hz
- Maksimum menzil = 4m
- Minimum menzil = 2cm
- Ölçüm açısı = 15^o(maksimum)

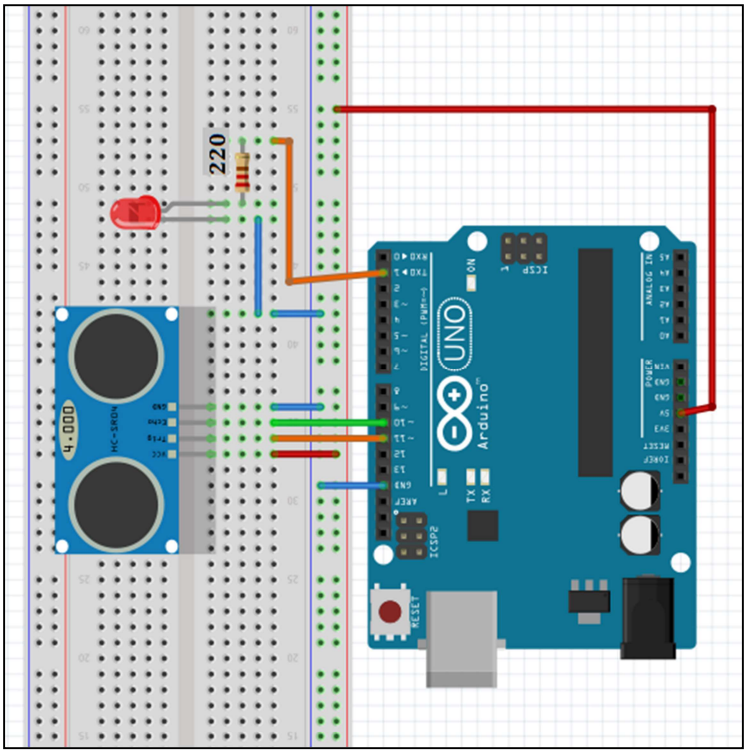
- Tetikleme sinyali süresi = 10us
- Ebatları 45*20*15 mm
- Ölçüm hassasiyeti = 3mm

Deney Malzeme Listesi:

- Arduino UNO
- 1 adet 220 Ω (ohm) direnç
- 1 adet led
- 2 adet HC-SR04 Ultrasonik Mesafe Sensörü
- Jumper
- Breadboard

Kod 7.8: Ultrasonik Sensör ile Mesafe Algılama

```
int tPin = 11;//sensörün trig pini
int ePin = 10;//sensörün echo pini
long sure;
long mesafe;//sensörden okuduğumuz uzaklık
void setup() {
  pinMode(1, OUTPUT);
  pinMode(tPin, OUTPUT);
  pinMode(ePin, INPUT);
  Serial.begin(9600);
}
void loop() {
  digitalWrite(tPin, LOW); //sensör ilk durumda deaktif
  delayMicroseconds(5);
  digitalWrite(tPin, HIGH); //sensörden ses dalgası yollandı
  delayMicroseconds(10);
  digitalWrite(tPin, LOW); //bir tek ses dalgası gönderildi
  sure = pulseIn(ePin, HIGH); //gönderilen dalganın geri
dönüş süresi
  mesafe = sure / 29.1 / 2; //geçen süreden mesafe hesabı
  Serial.println(mesafe);
  if (mesafe < 10)
    digitalWrite(1, HIGH);
  if (mesafe > 10)
    digitalWrite(1, LOW);
}
```


220
4.000
HC-SR04
UNO
Arduino

7.9 Deney #9 Servo Motor Kontrolü

Servo motorların çalışma mantığını daha önce anlatmıştık. Bu deneyimizde potansiyometreden aldığımız analog veriyle Arduino üzerinden Servo motor kontrolü yapacağız.

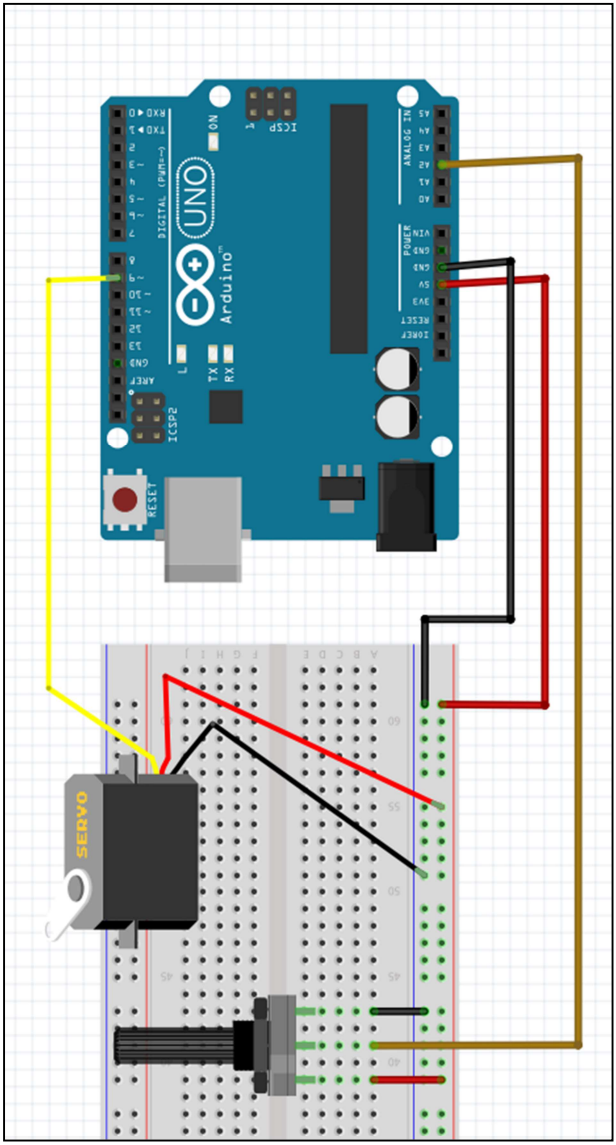
Deney Malzeme Listesi:

- 1 adet Pot
- 1 adet Servo Motor
- Arduino UNO
- Jumper
- 9V pil
- Breadboard
- Digital Multimetre

Kod 7.9: Servo Motor Kontrolü

```
#include <Servo.h>
Servo myservo;
int potPin = 2; //potun orta bacağının bağlı olduğu pin
int val;
void setup() {
myservo.attach(9); //motorun sarı kablosunun bağlı olduğu pin
}
void loop() {
val = analogRead(potPin);
 val = map(val, 0, 1023, 0, 179);
 myservo.write(val);
 delay(15);
  }
```

Servo motorumuzun sarı kablosun data kablosudur ve Arduinonun PWM pinlerinden birine bağlanır. RC Servo motorlar 0-180 derece arasında dönme hareketi yapma kabiliyetine sahiptirler ve bu dönme hareketini 1'er derecelik adımlar halinde yapabilirler. Bu yüzden de Potumuzdan aldığımız 0-1023 arası analog değerini 0-179 arasında bir değere oranlıyoruz. PWM hakkında daha geniş bilgi için Servo motoru anlattığımız bölümü tekrar inceleyebilir ve kitap sonundaki kaynaklara göz atabilirsiniz.


UNO
Arduino
SERVO

7.10 Deney #10 Flip-Flop Devresi

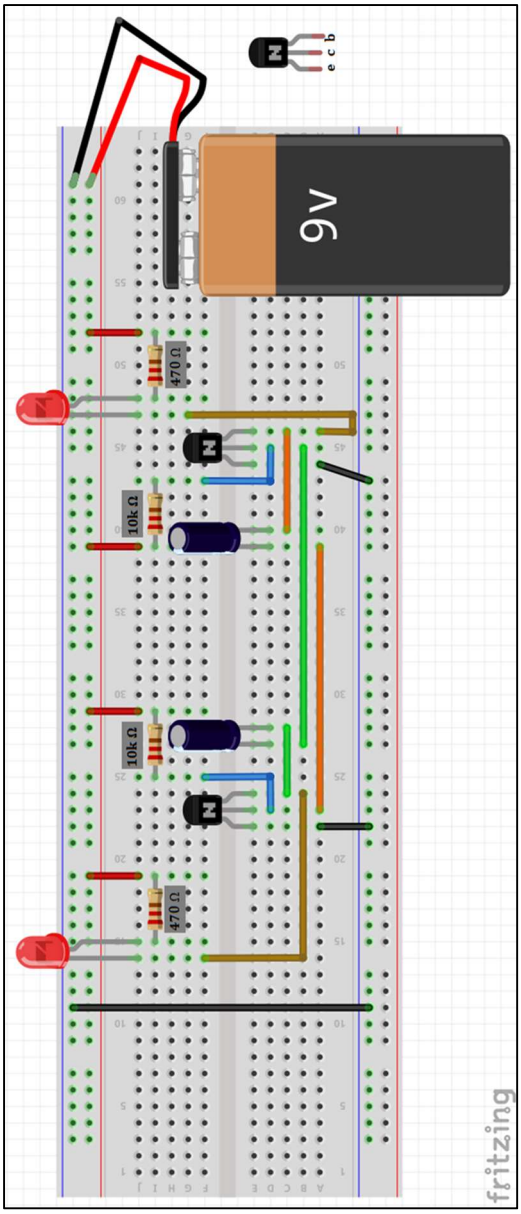
Bu deneyimizde Arduino kullanmadan bir devre kuracağız. Bu deney, elektronikteki temel devre elemanlarının işleyişlerini öğrenmek için çok iyi bir fırsat.

Deney Malzeme Listesi:

- 2 adet Led
- 2 adet 470 Ω direnç
- 2 adet 10k Ω direnç
- 2 adet BC 547 NPN Transistör
- 2 adet 100uF kondansatör
- Jumper
- 9V pil
- Breadboard
- Digital Multimetre

Bu deneyimiz öncekilerden biraz daha farklı olacak. Kontrol için yine ledlerden faydalanacağız ancak bu sefer Arduino gibi bir microcontroller'a ihtiyaç duymayacağız. Onun yerine transistör ve kondansatör kullanıp ledlerin sırayla yanmasını sağlayacağız. Çalışma mantığı ise şöyle: Devremize ledler üzerinden gelen 9V luk beslememiz transistörden geçer ve devreyi ilk kondansatör ve ikinci transistör üzerinden tamamlayıp, ledi yakar. Önemli bir bilgi, kondansatörler üzerinden geçen akımla dolarlar ve dolduktan sonra üzerlerinden akım geçirmezler. Bu yüzden kondansatör dolana kadar yanan led, daha sonrasında sönecektir. Bu sefer ilk transistör den geçmeye çalışan akım, ikinci transistör üzerinden biraz önceki yolu izleyerek devreyi tamamlar ve ikinci led yanar. Bu devir daimdir. Yani

kondansatörler sürekli dolup boşalırlar ve ledler bataryamız bitene kadar yanıp sönerler. Direnç ve kapasitör değerlerini değiştirerek ledlerin yanma sürelerini istenilen şekilde ayarlamak da mümkün.


9v
e c b
470 Ω
10k Ω
10k Ω
470 Ω
fritzing

7.11 Deney #11 Sıcaklıkla Fan Kontrolü

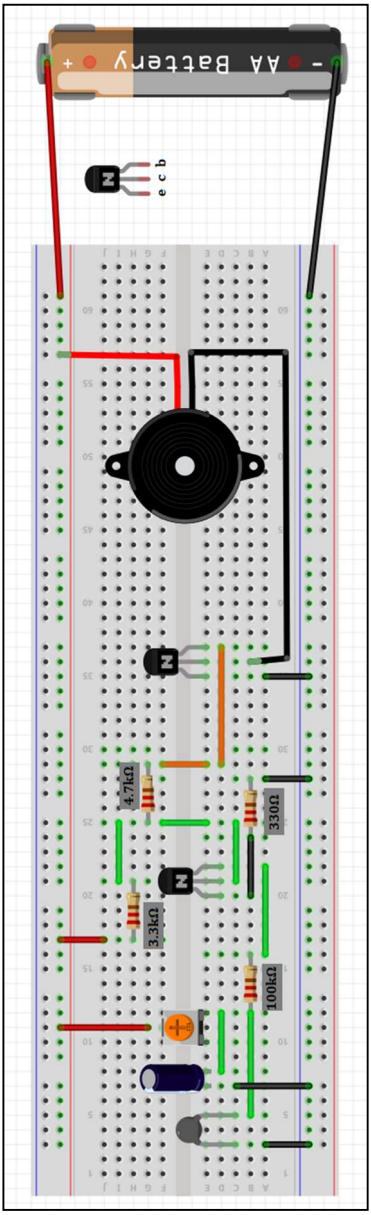
Bu deneyimizde NTC kullanarak ortamın sıcaklığının artışı ve azalışına göre fanımızı çalıştıracağız.

Deney Malzeme Listesi:

- 1 adet 10k NTC
- 1 adet 330 Ω (ohm) direnç
- 1 adet 4.7k Ω (ohm) direnç
- 1 adet 100k Ω (ohm) direnç
- 1 adet 3.2k Ω (ohm) direnç
- 1 adet 47k trimpot
- 1 adet 12V Fan
- 2 adet C945 NPN Transistör
- 22uF Kondansatör
- Jumper
- 12V pil
- Breadboard
- Digital Multimetre

Deneyin başında ledi, direnci, kondansatörü, transistörleri ve fanı, jumperlar ile üstteki deney şemasına uygun olarak breadboard üzerine yerleştirin. Bu deney aslında öğrenme deneyinden ziyade biraz daha uygulama deneyi. O yüzden kimi bağlantıları anlatmakta biraz güçlük çekilebilir. Bu tür devreleri yapmak zamanla kolaylaşır. Ve yine Arduino kullanmadan kendi kendini programlayan bir sistem olarak düşünebilirsiniz. NTC den alınan sıcaklık 'verisine' göre fanın dönme hızı artıp azalacak. Bilindiği üzere bahsedilen veri NTC' nin sıcaklık değişimine göre değiştirdiği direnç bilgisidir. (Fan yerine Arduino deneylerinde olduğu gibi yine ledler kullanılabilir. Ancak bu sefer batarya, kondansatör,

direnç değerleri azaltılarak ayarlanmalıdır. Aksi halde devreye fazla yükleme yapılarak led patlatılabilir.) Bu deneyi hakkını vererek yapabilirseniz, elektronik hakkındaki bilgi ve deneyiminizi gerçekten artıracak bir deney olacaktır. O yüzden bütün bağlantıları büyük bir dikkatle yapınız.


AA Battery
+
-
e c b
4.7kΩ
3.3kΩ
330Ω
100kΩ

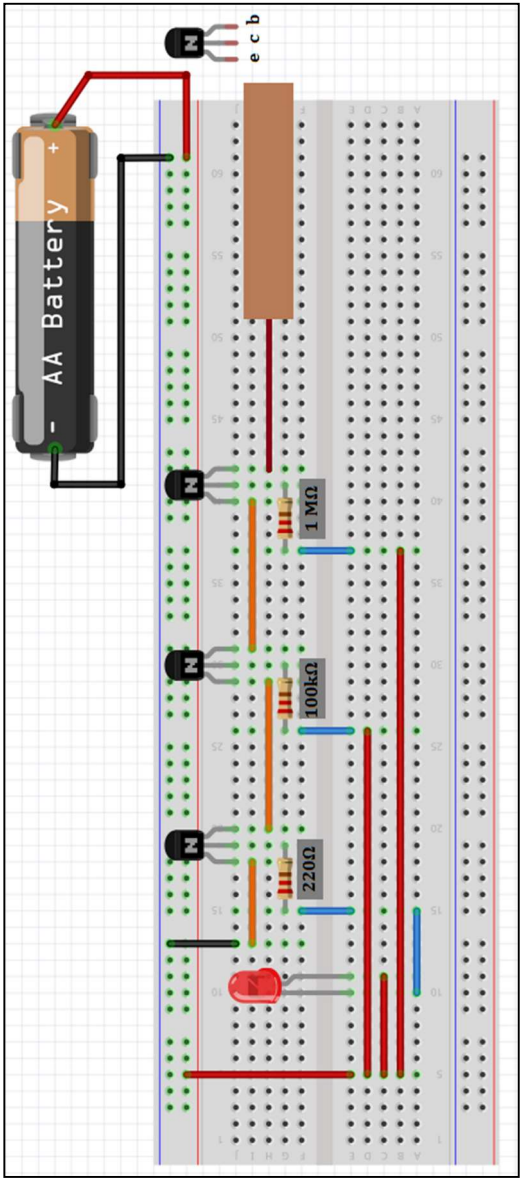
7.12 Deney #12 Statik Elektrik Algılama Aygıtı

Bu deneyimizde, transistörleri kullanarak kazanç artırmayı göreceğiz. Elektronikte kazanç bazen çok küçük miktarlarda akan akımın değerini manipüle edip yükseltmek, bazen de aynısını gerilim için yapmakta çok mühim bir yer tutar. Bu deneyde çevremizdeki statik elektriğin ayırdına varmış olacağız.

Deney Malzeme Listesi:

- 1 adet 220 Ω direnç
- 1 adet 1M Ω direnç
- 1 adet 100k Ω direnç
- 1 adet Led
- 3 adet BC547 NPN Transistör
- Jumper , Metal Levha
- 6V pil
- Breadboard

Digital Multimetre Her transistörün kendine özgü bir kazanç değeri vardır ve bu değer ß ile ifade edilir. Devremizde kullandığımız BC547 kodlu transistörün ortalama ß değeri 200'dür. Bu değer farklı transistör çeşitlerinde farklılık gösterdiği gibi aynı kod numarasına ve aynı fabrika üretimine sahip transistörlerde dahi değişkenlik gösterebilir. Devremizde 3 tane transistörü manipüle ederek emiter ve base kısımları birbirine bağlı olacak şekilde yerleştirdik. Bu sayede kazancımız her bir transistör için yaklaşık 200 olmak üzere genelde (200 x 200 x 200) yani 8x10^6 oldu. Bu kazanç sayesinde, metal levhayı statik elektrik olan bir yere tuttuğumuzda ledimiz artık yanmaya başlayacak. (bkz. Darlington Devresi)


e c b
AA Battery
+
-
1 MΩ
100kΩ
220Ω

MALZEME LİSTESİ

Malzeme İsmi	Adet
Arduino Uno	1
Jumper Kablo	30+
Direnç(150R, 220R, 330R)	20+
Direnç(4.7k)	20+
Potansiyometre(22k)	1
Direnç(10k, 100k, 1M)	20+
Trimpot(47k)	2
10k NTC	1
Transistör(BC547) NPN	3+
Transistör(C945) NPN	2
Kondansatör(22uf, 100uf)	3+
HC-SR04 Ultrasonik Mesafe Sensörü	1
DS18B20 Sıcaklık Sensörü	1
Breadboard	1
Digital Multimetre	1
Servo Motor	1
Fan 12V	1
Pil(9V)	1+
Buton	3+
Led	10+

YARARLI LİNKLER

1. http://www.robotiksistem.com
2. http://www.instructables.com/
3. http://electronics.stackexchange.com/
4. http://forum.arduino.cc/
5. http://fritzing.org/
6. https://www.arduino.cc/en/main/software#
7. https://oguzhanciftci.wordpress.com/
8. http://320volt.com/
9. http://www.robotiksistem.com/
10. http://www.ehlimuhendis.com/arduino/arduino-ya-yeni-baslayanlar-icin-4-basit-uygulama.html
11. http://yasincetin.net/hc-sr04-ultrasonik-sensor-ve-pic-16f877a-ile-mesafe-olcer/
12. http://kitap.akademikport.com/AkademikPort-Arduino-Baslangic-Projeleri.pdf?

Bu kitabın yazarı 1995 doğumlu ve Çanakkale, Biga'lı olan Oğuzhan ÇİFTÇİ, lise öğrenimini Bursa Milli Piyango Anadolu Lisesi'nde tamamladı. Marmara Üniversitesi Elektrik - Elektronik Mühendisliği Bölümünden 2018 senesinde mezun oldu. Şu anda Boğaziçi Üniversitesi Elektrik - Elektronik Mühendisliği Bölümünde Yüksek Lisans eğitimine devam ediyor. Java, C#(.NET) gibi OOP yanısıra C, Assembly x86, Verilog HDL ile de ilgileniyor ve projeler yapıyor. PCB devre kartı tasarımı programlarını da kullanıyor. Bosch Bursa firmasında bir süre Teknik Bakım bölümünde kısmi zamanlı olarak çalıştı. Devletimizin orta ve uzun vadede ihtiyacı olan bilimsel ve teknolojik gelişmelere katma değer sağlayabilmek için, aralarında Tübitak desteği de almış olan projeler üzerinde çalışıyor.
Yapılan güncel çalışmalar için daha geniş bilgi özgeçmişinde bulunabilir.

İletişim : a.oguzhanciftci at gmail.com