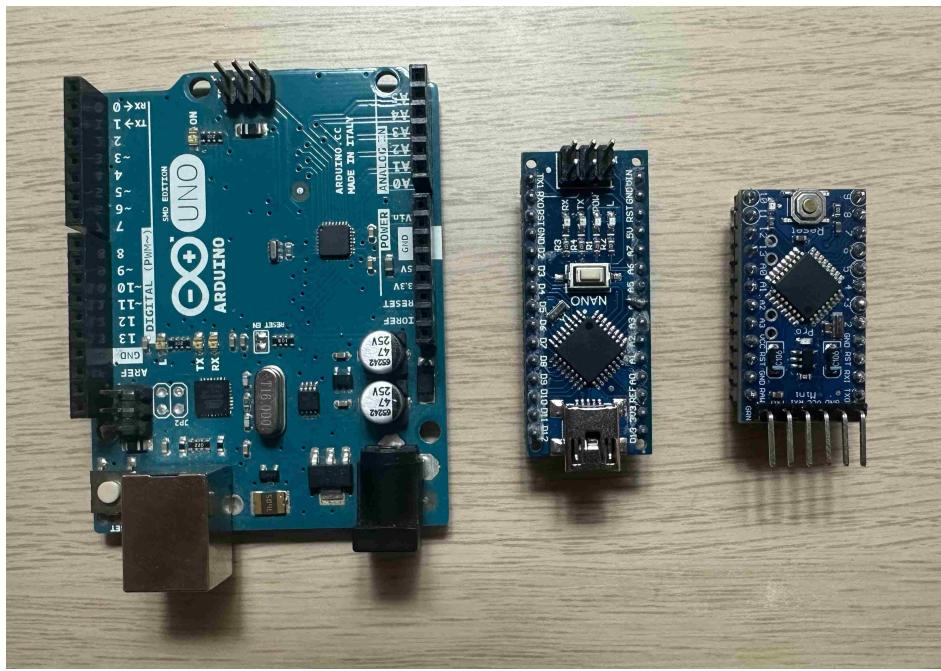


2. YAPILAN ARDUINO ÇALIŞMALARI

2.1. Arduino Hakkında Genel Bilgiler

Mühendislik çeşitli alt disiplinlerden (e.g., bilgisayar mühendisliği, elektrik-elektronik mühendisliği, makine mühendisliği, *vb.*) olılsa da mühendisler olarak bugün pratik bir proje yaptığımızda multidisipliner olarak çalışmak zorundayız. Sadece elektrik-elektronik mühendisliğinde değil neredeyse bütün alanlarda (hatta bazen sosyolojik meselelerde bile) ölçüm cihazı ve sensörler işin içeresine girdiğinde, yapılan hesaplamalarda bir beyin vazifesi gören hafif, küçük ve ucuz çipler olan mikrodenetleyicilere (microcontroller) ihtiyaç duyulmaktadır. Son on beş senede dünyada özellikle hobi projelerinde (*do it yourself - DIY*) en çok tercih edilen mikrodenetleyiciler Atmel şirketinin üretmiş olduğu 8-bit ATmega328 çipli Arduino'lardır. Kullanım kolaylığıyla teknik bilgisi olmayan kullanıcıları (e.g., sanatçilar) bile kendisine çekmeyi başaran Arduino, son zamanlarda yüksek işlem hızlarından dolayı 32-bitlik çipler (e.g., STM32) piyasaya sürülmüşne rağmen hâlen birçok uygulamada kullanılmaktadır. En yaygın ve popüler Arduino modelleri olan Uno, Nano ve Pro-Mini'yi Şekil 1'de görebilirsiniz. Bu çalışmanın hem tasarım hem bitirme bölümlerinde kurulan devrelerde her üç Arduino da kullanılmıştır.

Arduino'lar sensörlerden veri okumayı, bu verilerden alakalı değerleri hesaplamayı, ve bu parametreleri/değişkenleri görselleştirmeyi veya eyleyicileri sürekli sinyallere dönüştürüp (i.e., kapalı bir çevrim tasarlayıp) geri-beslemeli bir kontrol sistemi (e.g., robot) gerçeklemeyi muazzam kullanıcı desteği ve profesyonel dökümantasyona sahip yüksek seviyeli C++ API'si sayesinde inanılmaz kolaylaştırarak insanlığın gelişimine büyük katkıda bulunmuştur. Kendilerinden sonra gelen single board computer olarak geçen tek kartlı bilgisayarlar (e.g., Raspberry Pi, NVidia Jetson Nano) her ne kadar kullanım alanları daha çok bilgisayarlı görü (computer vision - CV), makine öğrenmesi (machine learning - ML) ve derin öğrenme (deep learning - DL) uygulamaları olsa da Arduino'ları tahtından edememiştir. Artık Arduino'lar da TinyML sayesinde sınırlı işlem gücüne sahip de olsa yapay zekâ (artificial intelligence - AI) uygulamaları koşturabilmektedir.



Şekil 1. Arduino Uno, Nano ve Pro Mini.

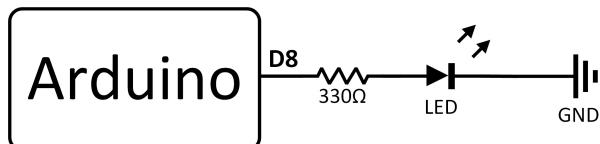
2.2. Arduino Projeleri

Bu bölümde EEM 216 dersinde yaptığımız örnek Arduino deneylerinin devre bağlantı şemalarını ve yazılımlarını bulabilirsiniz.

2.2.1. LED Yakma Devresi

Bu deneyde Arduino'nun dijital portunda tek bir bacak (pin) kullanacağız. Yukarıda Şekil 1'de ismi geçen Arduino modellerinde dijital port 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 olarak numaralandırılmıştır. İlk iki bacak olan D0 ve D1 pinlerinin üzerinde Rx ve Tx yazdigından genelde dijital port kullanan uygulamalar bu bacakları kullanmazlar. Burada D8 bacağını çıkış (output) olarak seçelim ve bir döngü içerisinde D8'i belirli bir süre yakalım ve söndürelim. Doğal olarak D8'i LED'e direkt değil bir direnç üzerinden bağlamalıyız ki LED'i yüksek akımdan dolayı yakmayıalı. Devre bağlantı şeması ve kodunu Şekil 2'te görebilir, koda ilgili kod deposundan¹ erişebilirsiniz.

¹<https://github.com/mtahakoroglu/gumushane-eem-kodlama>

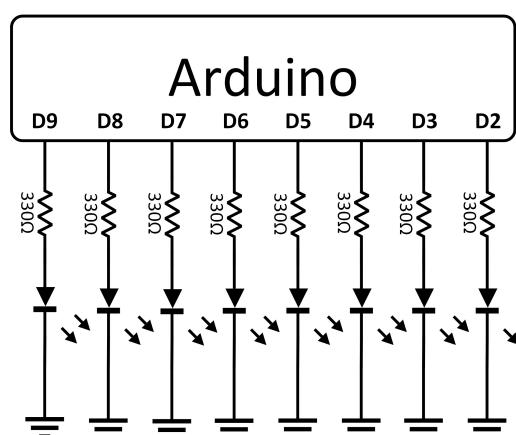


```
led_flash.ino
#define LED_PIN 8
void setup() {
    pinMode(LED_PIN, OUTPUT); // çıkış olarak seç
}
void loop() {
    digitalWrite(LED_PIN, HIGH); // LED'i yak
    delay(500); // ms cinsinden gecikme
    digitalWrite(LED_PIN, LOW); // LED'i söndür
    delay(500);
}
```

Şekil 2. LED yakma devresi.

2.2.2. Flaşör Devre

Sıradaki deneyde Arduino'nun dijital portunda sekiz bacağı birden çıkış olarak kullanacağız. Bu bacakları 2, 3, 4, 5, 6, 7, 8, 9 olarak seçelim. Algoritmik olarak önceki projeye göre farklı olan en önemli şey, kullanılan dijital pin sayısı birden fazla olduğu için bacakların kontrolünü **for** döngüleriyle gerçekleştiriyor olacağımızdır. Devrenin bağlantılarını ve kodunu Şekil 3'te görebilir, koda ilgili kod deposundan² erişebilirsiniz.



```
// D2'den D9'a kadar olan pinler, seçelim
int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9}; // 8 LED için pinler
int numLeds = 8; // LED sayısı
int f = 2; // frekans | saniyede kaç kere tur atsın
int delayTime = (1000/f)/(2*numLeds-2); // ms

void setup() {
    // LED pinlerini çıkış olarak ayarlıyoruz
    for (int i=0; i<numLeds; i++) {
        pinMode(ledPins[i], OUTPUT);
    }
}

void loop() {
    // LED'leri D2'den D9'a doğru yak
    for (int i=0; i<numLeds; i++) {
        digitalWrite(ledPins[i], HIGH); // LED'i yak
        delay(delayTime);
        digitalWrite(ledPins[i], LOW); // öbür LED'e geçmeden LED'i söndür
    }
    // LED'leri D9'dan D2'ye doğru geri yak (D9 ve D2 dâhil değil)
    for (int i=numLeds-2; i>0; i--) { // numLeds-2 çünkü D9 tekrar yanmamalı
        digitalWrite(ledPins[i], HIGH); // LED'i yak
        delay(delayTime);
        digitalWrite(ledPins[i], LOW); // öbür LED'e geçmeden LED'i söndür
    }
}
```

Şekil 3. Flaşör devre.

²<https://github.com/mtahakoroglu/gumushane-eem-kodlama>

2.2.3. Kare Dalga Üreteci

Buraya kadar Arduino'nun dijital portundaki pinleri **setup()** fonksiyonu içinde **pinMode()** fonksiyonuyla çıkış (**OUTPUT**) olarak seçmeyi ve ardından **loop()** fonksiyonu içinde **digitalWrite()** fonksiyonuyla seçili bacaklarda çıkış olarak logic 0 (**LOW**) ve logic 1 (**HIGH**) sinyalleri üretmeyi başardık. Bu yeni deneyde dijital portu çıkış olarak seçmeye ek olarak analog giriş kullanacağız.

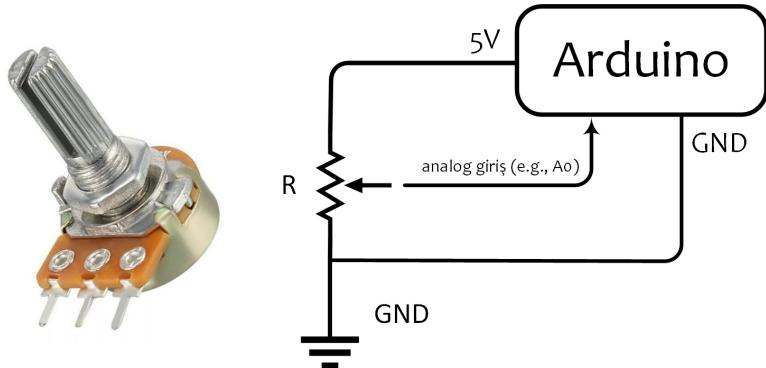
Ayarlı direnç olarak bilinen potansiyometre kullanarak Arduino'ya analogdan dijital dönüştürücü yardımıyla transfer edeceğimiz [0-5]V aralığındaki gerilim değeriyile D13 dijital pininde üreteceğimiz bir kare dalga sinyalinin frekansını (periyyodunu) değiştireceğiz ve sinyal jeneratörü yapacağız. Oluşturduğumuz bu sinyali hem osiloskoba hem hoparlöre (veya buzzer'a) bağlayarak görsel ve işitsel olarak da gözlemleyeceğiz.

Devrelerde gerilim bölücü (voltage divider) olarak kullanılan potansiyometre;

- Üç bacağa sahiptir.
- Uç bacakları (yâni orta bacak dışındakiler) sıra gözetmeksizin Arduino'nun Vcc (5V) ve GND pinlerine bağlandığında orta bacağı [0-5]V aralığında analog bir değer üretir.
- Orta bacağı Arduino'ya analog girişlerden biriyle (e.g., A0) bağlandığında analogdan dijital çevirici (analog to digital converter - ADC) aracılığıyla [0-5V] aralığında sürekli (continuous) zamanlı analog sinyal, [0-1023] aralığında bir tam sayıya (integer) doğrusal olarak dönüştürülerek (linear mapping) ayrık (discrete) bir değer elde edilir. Artık Arduino'da bu tam sayı değeri kullanılarak birçok şey (e.g., joystick ile kanal/motor sinyali üretme) yapılabilir.

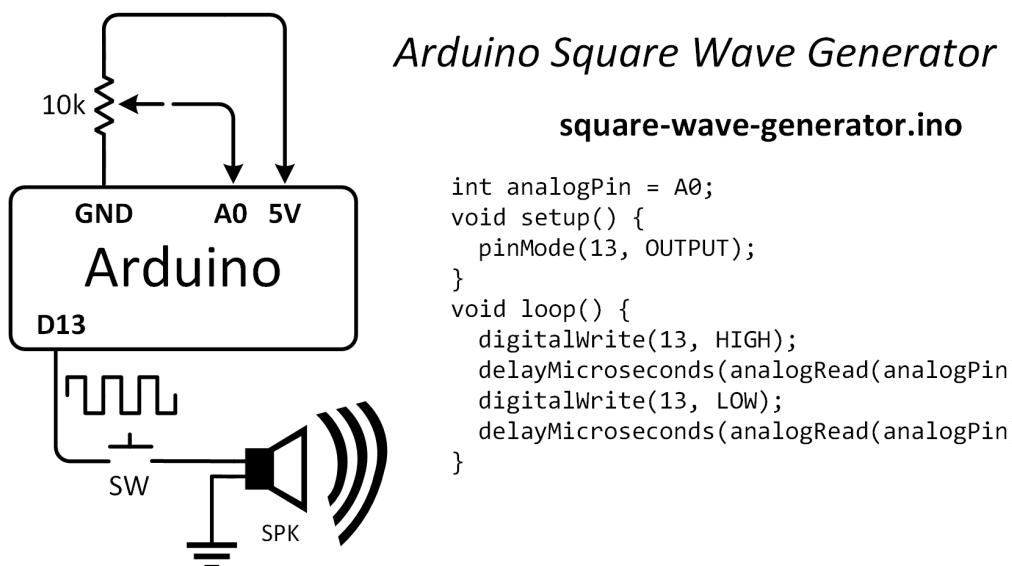
Bilindiği gibi potansiyometreler, iç mekân ışıklandırma sistemlerinde ışık şiddetini ayarlama, elektronik aletlerde ses düzeyini değiştirme, RC model araba/uçak kontrol eden uzaktan kumandalarda kanal sinyallerini oluşturma gibi pek çok önemli uygulamada vazife yapmaktadır. Örnek bir potansiyometreyi Şekil 4'de görebilirsiniz.

En sol ve en sağ bacakları daha önce de belirttiğimiz gibi sıra farketmeksizin GND ve 5V'a, orta bacağı ise Arduino'nun A0 analog giriş pinine Şekil 4'deki gibi bağlayıp yukarıda bahsedilen ADC işlemini gerçekleştireceğiz.



Şekil 4. Örnek bir potansiyometre ve Arduino bağlantıları.

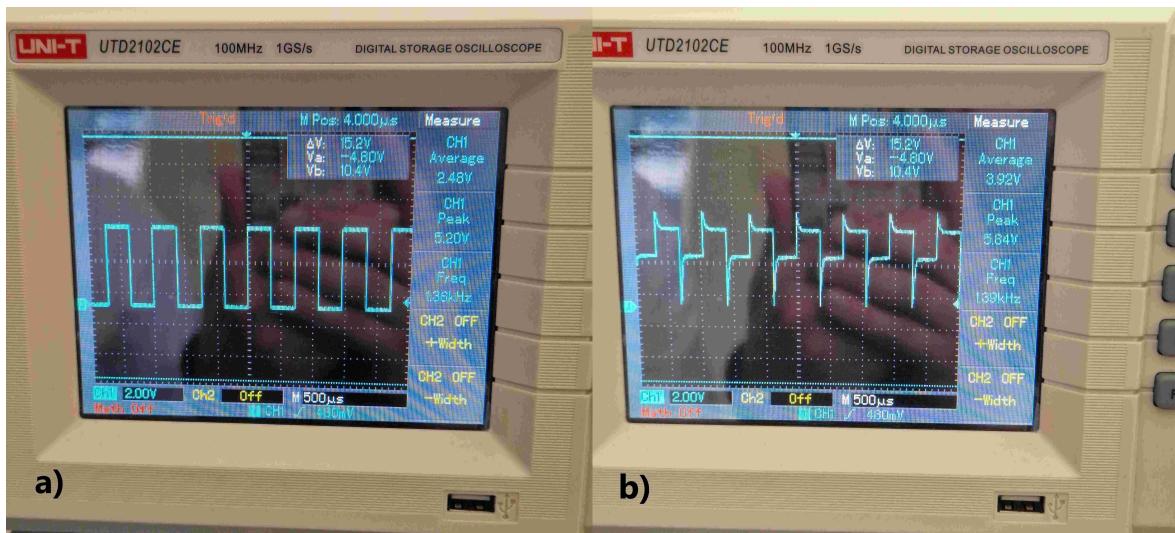
Buradaki deneyde D13 pininde ürettiğimiz sinyali hoparlör veya buzzer'a üstten basmalı anahtar (push-button switch) üzerinden bağlamak isabetli olacaktır zira sürekli sinyalin oluşturduğu sesi duymak istemiyoruz. Yâni anahtara bastığımız zamanlarda sesi duyacağız. Şekil 5'de kare dalga üreteci devre şemasında devre elemanları arasındaki bağlantıları ve Arduino kodunu görebilirsiniz.



Şekil 5. Kare dalga üreteci devre şeması ve kodu.

Şekil 5'te D13 no'lu dijital pinde üretilen kare dalga sinyalini osiloskopun probuna bağlayıp da gözlemlersek hoparlör/buzzer aracılığıyla duyduğumuz seste potansiyometre aracılığıyla meydana gelen frekans/periyot değişimini Şekil 6'daki gibi gözlemleriz. Şekil 6'te a) ile gösterilen sinyal, üstten basmalı anahtar açık devreyken (yâni anahtara basılmadığında) görüntülenen sinyaldir. Bu durumda hoparlör/buzzer devreye yük oluşturmadığından akım çekmeyecek ve dolayısıyla gerilim düşümüne (güç transferine) sebebiyet vermeyecektir. Sonuç olarak D13'de üretilen kare dalga sinyali aynen teorikte

olduğu gibi mükemmel bir biçimde (yâni bozulmadan) gözlemlenebilmiştir. Öte yan- dan push-button SW'e basıldığı anda devre tamamlandığı için hoparlör/buzzer akım çekecek ve oluşan gerilim düşümü (güç transferi) sonucu b) ile gösterilen şeke sahip bir kare dalga oluşacaktır. Anahtara basıldığında sinyalde gözlemlenen bu bozulmanın yaşanmaması için transistörler veya işlemel yükselteç (operational amplifier yâni op-amp) yardımıyla tampon (buffer) devresi kurularak D13 ve hoparlör/buzzer arasına konabilir.



Şekil 6. Kare dalga sinyalinin osiloskop ekranındaki görüntüsü.

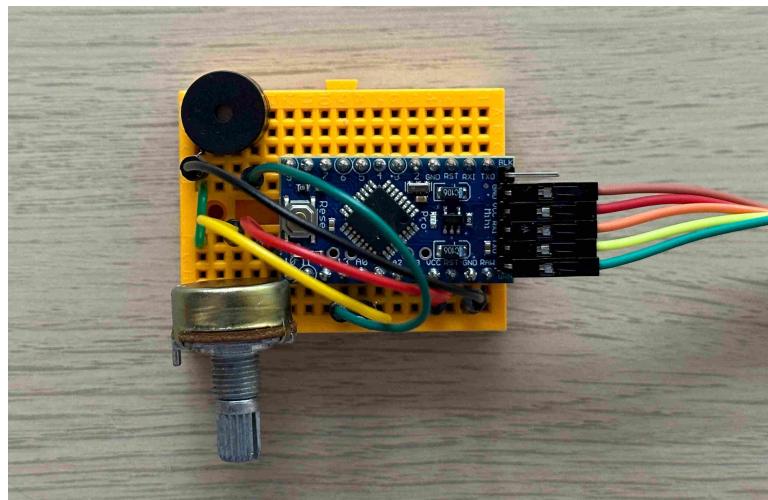
Kare dalga sinyali üretici deneyinin Arduino ve devre elemanlarıyla kurulmuş hâlini Şekil 7'de görebilirsiniz. Yapılan deneyin videosuna da ilgili kod deposundan ulaşabilirsiniz.³ Burada Yapay Zekâ'dan faydalanan Python'da **matplotlib** paketi kullanılarak bilgisayar ekranında osiloskop işlevi görecek bir kod oluşturulabilir.

2.2.4. Dijital Giriş Okuma

Bu bölümde ikinci deney olarak yaptığımız flaşör devre çalışırken üstten basmalı anahtar yardımıyla D10 pininden dijital bir sinyal okuyup animasyon hâlinde yanan LED'leri durdurmak veya donmuş hâldeki LED animasyonunu devam ettirmek istiyoruz. Bunu yapmak için D10 pinine ait **pull-up** rezistörünü **setup()** fonksiyonunun içinde

```
pinMode(buttonPin, INPUT_PULLUP);
```

³<https://www.youtube.com/shorts/tjAw2EqMBL8>



Şekil 7. Kare dalga sinyali jeneratörü devresi.

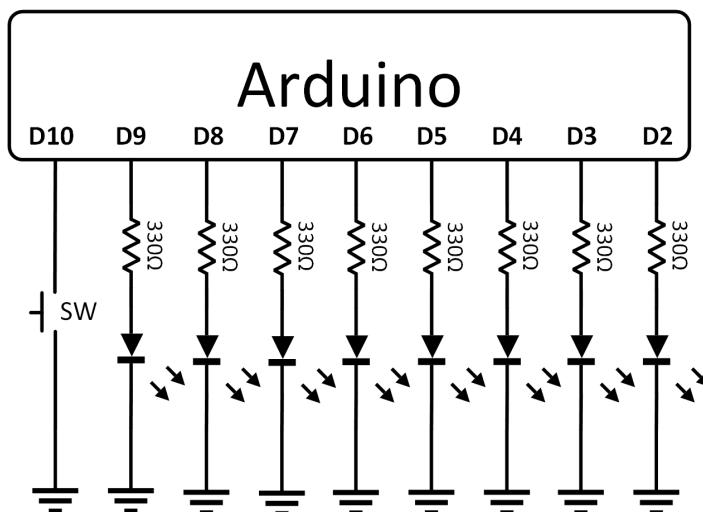
fonksiyonunu çağrıarak aktif hâle getirmek ve ardından **loop()** fonksiyonunun hemen ilk satırında

```
digitalRead(buttonPin)
```

fonksiyonu çağrıarak sürekli D10'dan gelecek olan sinyali Arduino'ya okumamız lâzım.

Burada **buttonPin** isimli değişken 10 olarak önceden tanımlanıyor (yâni D10 pini).

Şekil 8'da devre bağlantılarını görebilir ve kodun tamamını aşağıda bulabilirsiniz.



Şekil 8. Kare dalga sinyali jeneratörü devresi.

```
1 const int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
2 const int numLeds = 8;
3 const int buttonPin = 10;
4
```

```

5 int currentLed = 0; // Şu anki LED konumu
6 int direction = 1; // 1: ileri, -1: geri
7 bool flashing = true; // LED yanıp sonme durumu
8 int buttonState; // Mevcut buton durumu
9 int lastButtonState = HIGH;
10 unsigned long lastDebounceTime = 0;
11 const unsigned long debounceDelay = 50;
12 int f = 2;
13 unsigned long delayTime = (1000 / f) / (2 * numLeds - 2);
14 unsigned long previousMillis = 0;
15
16 void setup() {
17     for (int i = 0; i < numLeds; i++) pinMode(ledPins[i], OUTPUT);
18     pinMode(buttonPin, INPUT_PULLUP);
19     buttonState = digitalRead(buttonPin);
20 }
21
22 void loop() {
23     int reading = digitalRead(buttonPin);
24
25     // Debounce kontrolü
26     if (reading != lastButtonState) {
27         lastDebounceTime = millis();
28     }
29
30     if ((millis() - lastDebounceTime) > debounceDelay) {
31         if (reading != buttonState) {
32             buttonState = reading;
33             if (buttonState == LOW) {
34                 flashing = !flashing; // Butona basıldığında durdur/baslat
35             }
36         }
37     }
38
39     lastButtonState = reading;
40
41     // Eğer flashing aktifse ve zaman dolduysa LED güncelle
42     if (flashing && (millis() - previousMillis >= delayTime)) {

```

```

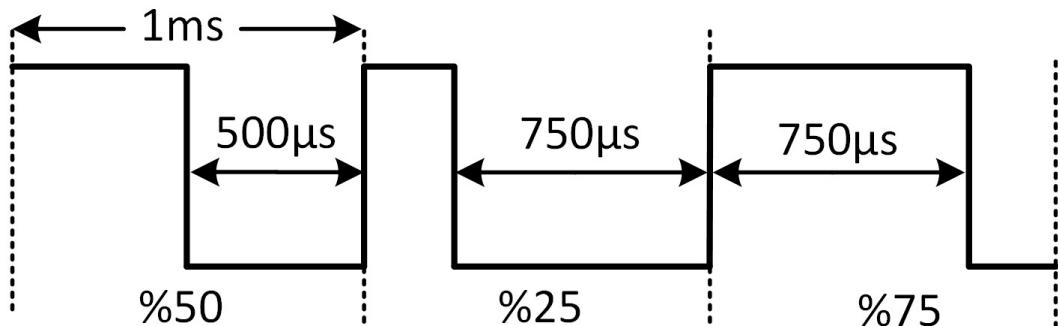
43     previousMillis = millis();
44
45     // Onceki LED'i sondur
46     for (int i = 0; i < numLeds; i++) digitalWrite(ledPins[i], LOW);
47
48     // Su anki LED'i yak
49     digitalWrite(ledPins[currentLed], HIGH);
50
51     // Sonraki LED'i belirle
52     currentLed += direction;
53
54     // Uclara ulastiginda yon degistir
55     if (currentLed >= numLeds) {
56         currentLed = numLeds - 2;
57         direction = -1;
58     } else if (currentLed < 0) {
59         currentLed = 1;
60         direction = 1;
61     }
62 }
63 }
```

2.2.5. PWM Sinyali ve Duty Cycle

Şu ana kadar dijital portun bacaklarını çıkış olarak seçmeyi **digitalWrite()**, giriş olarak seçmeyi **digitalRead()** ve [0-5]V aralığından analog bir sinyali Arduino'ya giriş olarak (tam sayı şeklinde) okumayı da **analogRead()** fonksiyonu ile gerçekleştirdik. Dijital ve analog port isminin direkt olarak geçtiği komutlardan sadece **analogWrite()** fonksiyonunu kullanmadık.

Robotikte karşımıza çıkan motorlardan bazıları DC motor, adım (step) motor, servo motor ve fırçasız (brushless) motorlardır. DC motorları sürmek için gereken akım veya gerilim değerini analog olarak üretmek yerine bu etkiyi oluşturacak (aynen Devre 2 Lab. dersinde gördüğümüz etkin değer hesabı gibi) bir dijital sinyal oluşturulmak yaygındır. Darbe Genişlik Modülasyonu (Pulse Width Modulation - PWM) denilen bu sinyalde logic 1 (+5V) süresinin logic 0 (GND) süresine oranını (periyodun içinde olarak

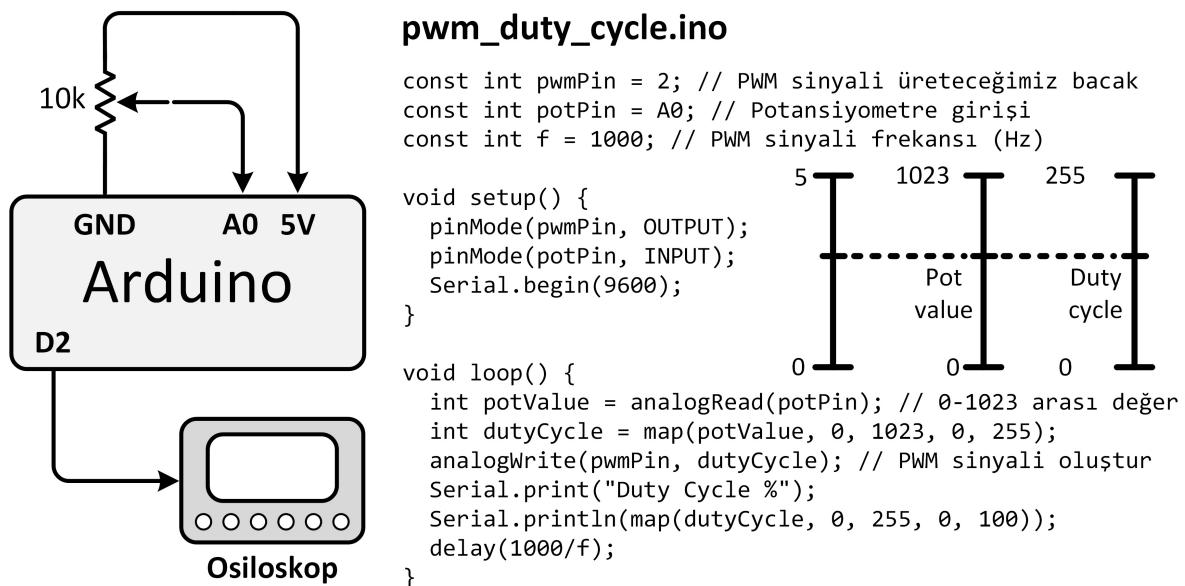
düşünün) artırıp azaltarak motora uygulanan voltajın ağırlıklı ortalamasını değiştirebilmekte ve böylece motorun dönüş hızını kontrol edebilmektedir. Frekansı 1kHz olan (yâni periyodu 1ms) örnek bir PWM sinyalini ve farklı duty cycle değerlerini Şekil 9'da görebilirsiniz.



Şekil 9. PWM sinyali değişken duty cycle değerleri.

İlgili devre ve kod Şekil 10'da görülmektedir. Kodu çalıştırdıktan sonra osiloskop yoksa bile seri port ekranını açıp PWM sinyalinin duty cycle yüzdesini gözlemlleyebilirsiniz.

Arduino PWM Signal Generation – Duty Cycle



Şekil 10. PWM sinyali oluşturma (`analogWrite()` ile).

2.2.6. Potansiyometre ile Servo Motor Kontrolü

Remote Control (RC) ile kontrol edilen oyuncaklar ve cihazlarda (e.g., model uçak, araba, drone) sıkıkla kullanılan servo ve fırçasız (brushless) motorları kontrol etmek için Arduino'nun zamanlayıcılarını (timers) kullanarak özel olarak oluşturduğu darbe genişlik modülasyonu (pulse width modulation - PWM) denilen bir metot var. Buradaki PWM sinyalinin yapısı yukarıda **analogWrite()** fonksiyonuyla ürettiğimiz sinyalin yapısından farklı. Bazı yerlerde **kanal sinyali** olarak da geçiyor. Arduino kullanıcıları bu sinyali kolayca oluşturmak için özel bir kütüphane kullanıyorlar: **Servo**. Arduino üzerindeki dijital pinlerden 3, 5, 6, 9, 10, 11 no'lu olanlar (Arduino üzerinde ~ işaretinden anlayabilirsiniz) Servo kütüphanesi kullanarak PWM (kanal) sinyali üretmek için kullanılabilir. Şekil 11'de D9 pininin PWM çıkışının seçildiğini görebilirsiniz. Burada servo motorun açısal konumunu kontrol ediyoruz. Bunun için potansiyometreyi A2 girişinden

```
int x = analogRead(A2)
```

komutuyla okuyarak $[0-5]V \rightarrow [0-1023]$ dönüşümünü yapıyoruz. Ardından Şekil 11'da da görülen

```
int y = map(x, 0, 1023, 0, 180)
```

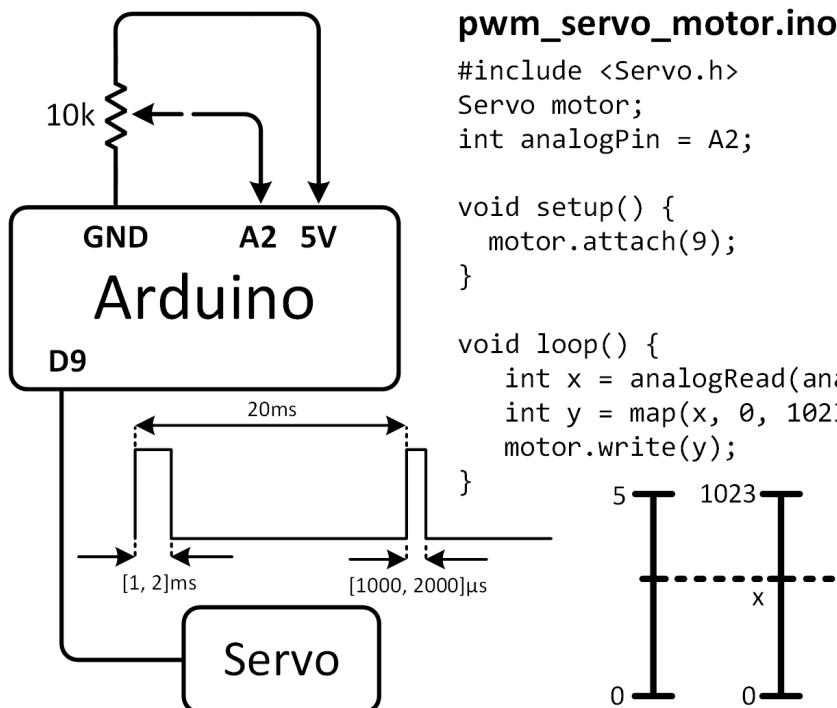
doğrusal dönüşümüyle servo motora uyguluyoruz.

Şekil 11'deki devre bağlantılarında görüldüğü gibi servo motorun kablolarından kahverengi olan Arduino'nun GND bacağına, orta kırmızı kablo +5V bacağına ve son olarak da turuncu kablo Arduino D9 bacağından üretilen PWM sinyaline bağlanıyor.

2.2.7. Joystick ile Fırçasız Motor Kontrolü

Bu kısımda bir önceki deneyde kullandığımız Servo kütüphanesi ile servo motora uyguladığımız PWM sinyalinin aynısını üreterip drone'larda yaygın olarak tercih edilen bir fırçasız motora (brushless motor) uygulayacağız. Ancak bu sefer bu işi RC alıcı-verici (receiver-transmitter) devrelerinde karşımıza çok çıkan bir kavram olan **PWM sinyalinin ms veya us olarak nitelendirilmesi** kavramını öğrenmek amacıyla osiloskop ekranında PWM sinyalini gözlemleyerek yapacağız. Osiloskoba erişimin olmadığı durumlarda, üretilen PWM sinyalinin nümerik değerini kontrol edebilmek için sinyale

Arduino Servo Motor Control



Şekil 11. PWM sinyali ile servo motor kontrolü.

tekabül eden değeri mikrosaniye cinsinden seri port ekranına da yazdıracağız. Elimizde bulunan eski bir drone kumandasının içine bir tane Arduino Pro Mini koyduk. Drone kumandasında dört adet potansiyometre var. Yâni iki adet joystick ediyor. Bunlardan birisi drone'u aşağı-yukarı hareket ettirmek için kullanılan **THROTTLE** denilen kanal. Biz bu kanala tekabül eden potansiyometreyi yukarıda servo motor ko-dunda olduğu gibi A2 girişinden okuyup ardından yine benzer bir doğrusal dönüşümle ama bu sefer $[0.9-2.1]\text{ms} = [900-2100]\mu\text{s}$ aralığına çevireceğiz.

```
int z = map(x, 0, 1023, 900, 2100)
```

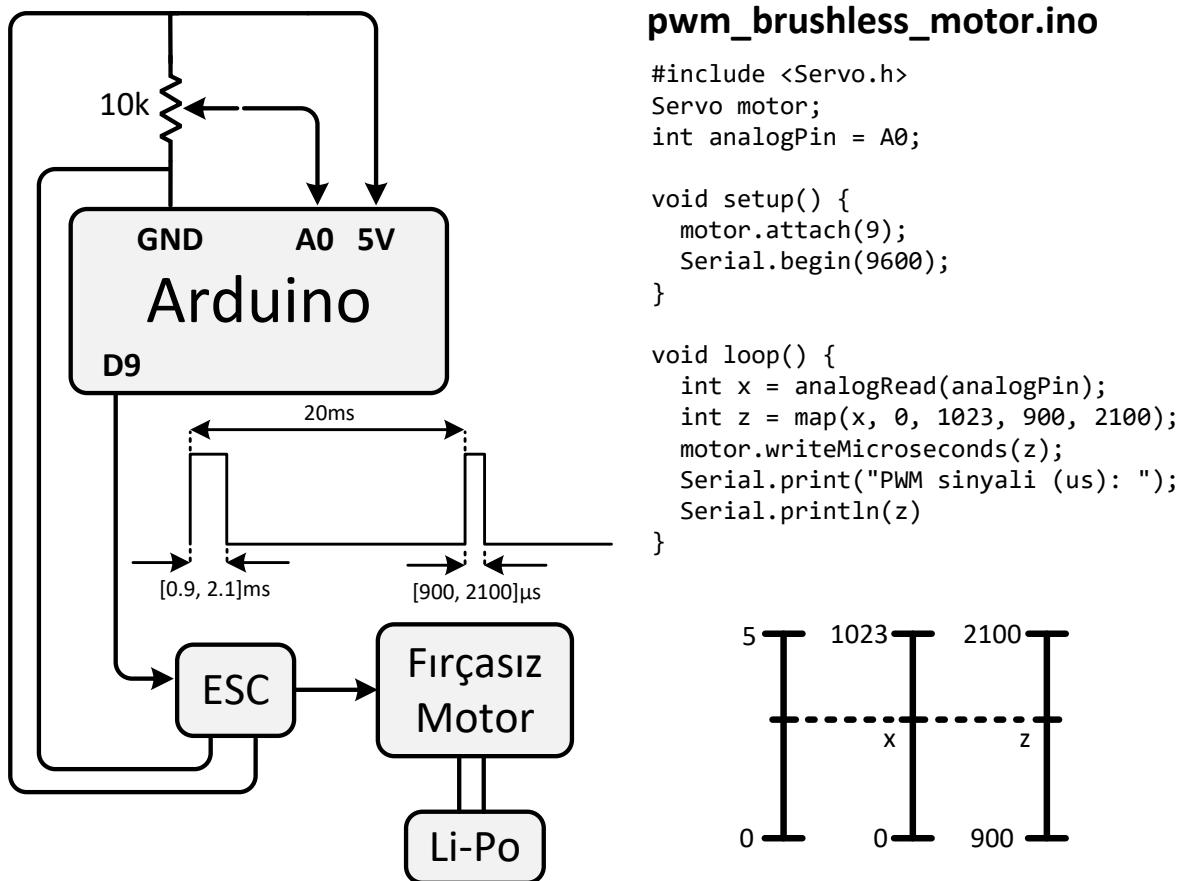
Son olarak da kanal sinyalini mikrosaniye birimi cinsinden oluşturacağız.

```
motor.writeMicroSeconds(z)
```

Böylece servo motora uyguladığımız PWM sinyalinin benzerini bu sefer bir fırçasız motorun ESC denilen motor sürücü devresine bağlayarak joystick veya potansiyometre aracılığıyla fırçasız motorun hızını kontrol edeceğiz. Şekil 12'de devre bağlantılarını ve kodu görebilirsiniz.

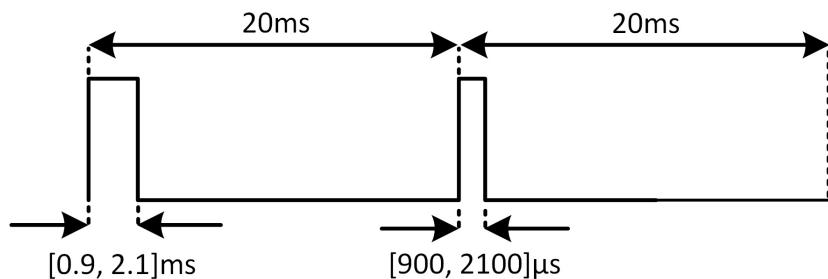
Servo kütüphanesiyle hem servo motor için hem fırçasız motor için PWM sinyali ürettiğimizde aslında sinyalin formu her zaman Şekil 13'deki gibi olmaktadır. PWM

Arduino Brushless Motor Control



Şekil 12. PWM sinyali ile fırçasız motor kontrolü.

sinyalının minimum ve maksimum değerleri bazı alıcı-vericilerde [1000-2000]us aralığında (e.g., Flysky), bazlarında [1100-1900]us aralığında (e.g., Spektrum), bazlarında ise [900-2100]us aralığında olabilmektedir.⁴ Kullanılan alıcı-verici markasına göre uygun aralıkta sinyal üretmek gerekmektedir.



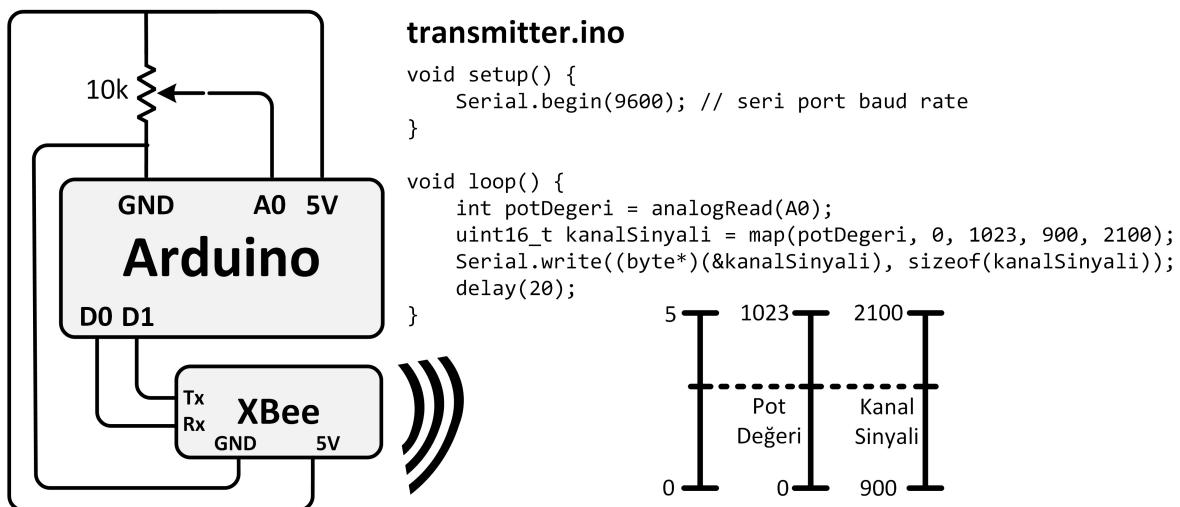
Şekil 13. Servo ve fırçasız motor kontrolü için üretilen örnek bir PWM sinyali.

⁴Bu değerler servo motor için PWM sinyali üretmede kullandığımız **write()** fonksiyonunda daha farklı bir aralıktaydı.

2.2.8. XBee Kablosuz Modül ile Fırçasız Motor Kontrolü

Bir önceki deneyde potansiyometre (veya joystick) ile okuduğumuz analog voltaj değerlerini kanal sinyali hâline getirip Arduino'nun PWM sinyali üretebilen bacaklarından biri olan D9 üzerinden fırçasız motora bağlı Electronic Speed Controller (ESC) ünitesine uygulamış ve Li-Po batarya ile güç verdigimiz fırçasız motor ve pervane ikili-sinin hızını kontrol etmiştim. Burada aynı işi bu sefer kablosuz (wireless) olarak yapmak istiyoruz. Bunun için potansiyometreden veri okuyup kanal sinyali hâline getirdiğimiz kısmını verici (transmitter), kanal sinyalinden PWM sinyalini oluşturup ESC'ye uygunladığımız kısmını da alıcı (receiver) olarak ayarlayacağız. Ardino'lara kolayca entegre olabilen XBee kablosuz RF modül Arduino'nun seri portu ile uyumlu çalıştığından kablosuz veri gönderip almada kodumuz karmaşık hâle gelmeyecek.⁵ Aynı sadelikteki bir alıcı ve verici koduyla bu işi yapacağız. Şekil 14'te verici devresinin bağlantılarını ve kodunu, Şekil 15'te ise Li-Po batarya ile güç verilerek çalıştırılan bir Arduino Nano ve XBee verici devresinin fotoğrafını görebilirsiniz.

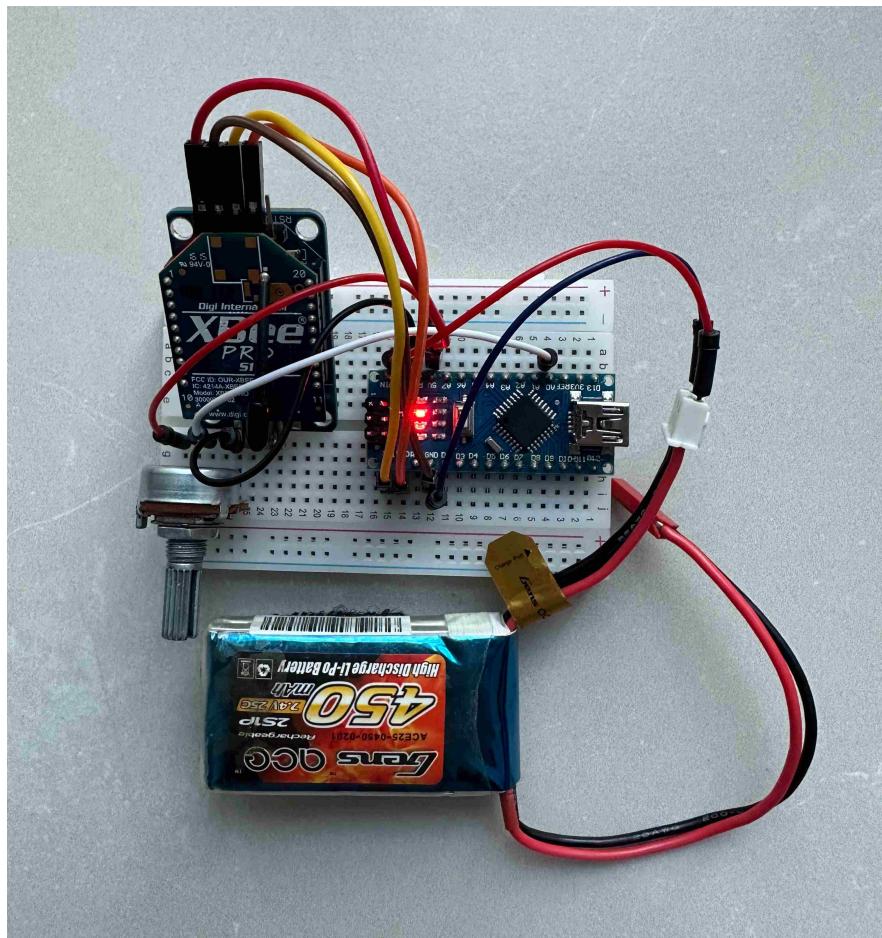
XBee Wireless Brushless Motor Control - TRANSMITTER



Şekil 14. Kablosuz fırçasız motor kontrolü verici devresi bağlantıları ve kodu.

Şekil 14'teki konfigürasyonda XBee kablosuz modülünden veri göndermek için seri port kullanıldığından Şekil 15'te mecburen Li-Po batarya ile devreye güç verdik. Eğer USB ile güç verseydik devrenin XBee'den kablosuz biçimde kanal sinyalini yollaması

⁵Kablosuz haberleşmeyi XBee'lere göre çok daha ucuz olan nRF24l01 modülü ile yaparsanız kodun daha karmaşık hâle geldiğini görebilirsiniz. Ayrıca pratikte nRF24l01'ler XBee'lere kıyasla daha fazla sorun çıkarmaktadır. Yâni performans olarak XBee'ler nRF24l01'lere göre daha üstündür.



Şekil 15. Verici (transmitter) devresi.

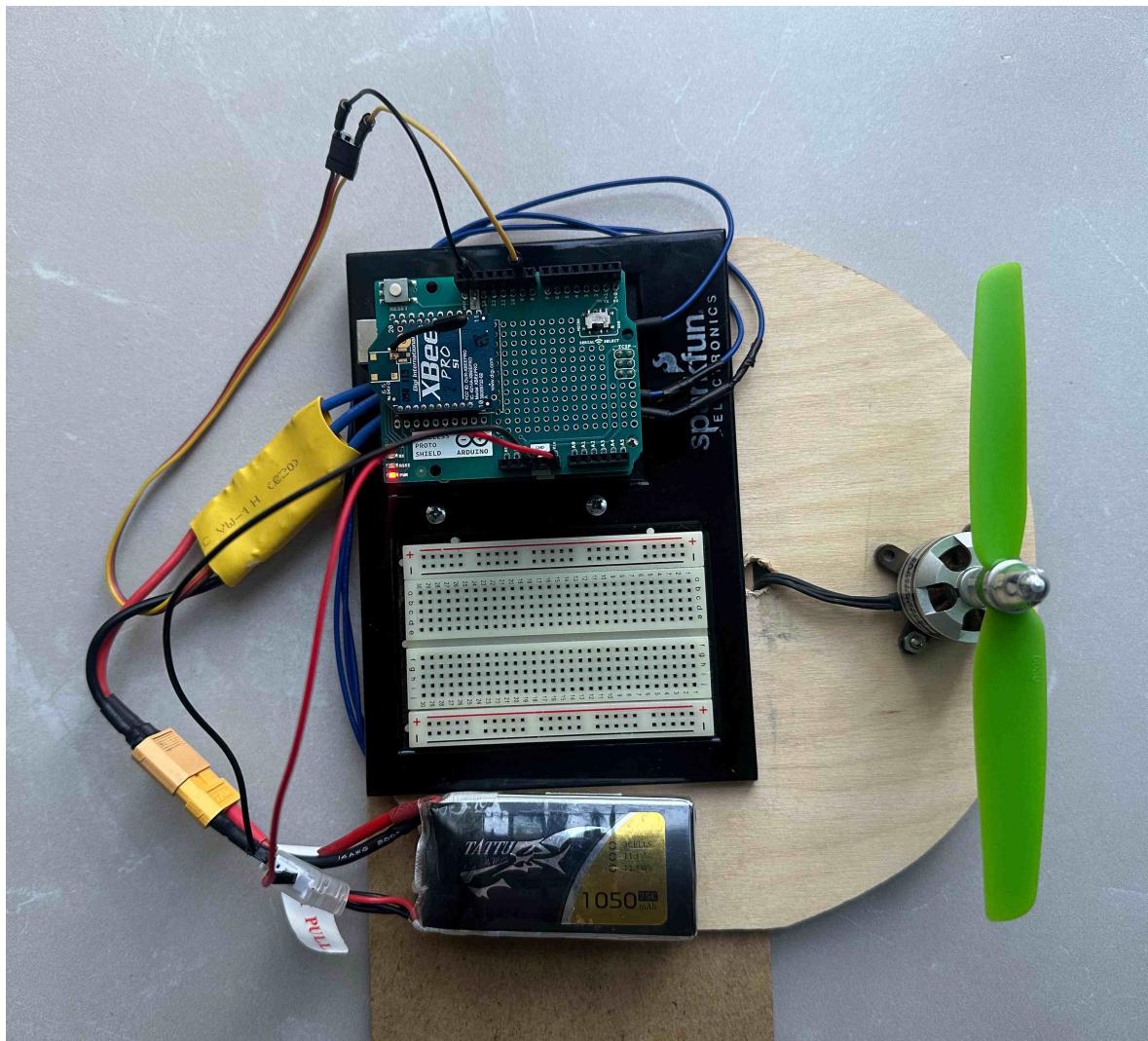
İçin bu sefer Software Serial olan olarak bilinen kütüphaneyi kullanmamız gerekecekti ki bu işlem kodu biraz daha uzun yapacaktı. Ayrıca Şekil 14'teki D0 ve D1 pini bağlantılarını D2 ve D3 olarak değiştirmemiz de gerekecekti. Bu yüzden yukarıdaki konfigürasyonda kalacağız.

Alicı (receiver) devresini ise bizim kurmamız gerekmiyor. Yine de burada bir bütünlük arz etmesi açısından onu da inceleyelim. Burada kullandığımız alıcı devresinde Arduino Uno kullandığımızdan dolayı XBee shield kullanarak XBee kablosuz modülü Arduino'ya bağladık. Fırçasız motoru süren ESC devresinin PWM sinyali bağlantısını yine önceki deneylerde olduğu gibi D9 no'lu pin üzerinden yapıp gücünü üç hücreli bir Li-Po batarya üzerinden sağlayacağız. Burada belirtmemiz gereken önemli bir pratik nokta var: Li-Po batarya ESC'ye bağlanınca ESC'nin sinyal kablolarından orta kablo olan kırmızı kablo otomatik olarak +5V olduğundan dolayı bu orta kabloya Ardu-

ino'dan ekstra olarak +5V bağlantısı çekmeye gerek yok.⁶. Son olarak da XBee shield bağlanmış Arduino Uno'muzu Vin bacağından Li-Po batarya'nın dengeleyici (balancer) kısmındaki gerilimden beslemek zorundayız. Eğer Li-Po bataryamız 2-3-4 hücreli ise bunda bir sorun olmayacağından emin olabiliriz ama daha az veya yüksek hücreli Li-Po'larda Arduino veya batarya zarar görebilir. Şekil 16'te Li-Po batarya ile güç verilerek çalıştırılan bir Arduino Uno, XBee kablosuz modülü (shield ile bağlanmış hâlde) ve ESC tarafından sürülen fırçasız motor - pervane ikilisinin fotoğrafını görebilirsiniz. Ayrıca aşağıda alıcı kodunu da bulabilirsiniz.

```
1 #include <Servo.h>
2
3 Servo motor;
4 const int motorPin = 9;
5 uint16_t kanalSinyali;
6
7 void setup() {
8     Serial.begin(9600);
9     motor.attach(motorPin);
10    motor.writeMicroseconds(900); // sinyal yoksa bunu uygula
11 }
12
13 void loop() {
14     if (Serial.available()) {
15         Serial.readBytes((byte*)&kanalSinyali), sizeof(kanalSinyali));
16         motor.writeMicroseconds(kanalSinyali);
17     }
18 }
```

⁶Servo motor kontrolü deneyinde orta kırmızı bacağa +5V bağlamıştık.



Şekil 16. Alıcı (receiver) devresi ve firçasız motor.

3. SENSÖRLER

Birçok mühendislik uygulamasında (e.g., otonom arabalar) mesafe ölçümü yapmak gerekmektedir. Bu iş için en çok tercih edilen sensörler olarak GPS (Global Positioning System), LIDAR (Light Detection and Ranging), SONAR (Sound Navigation and Ranging), IMU (Inertial Measurement Unit), kızılıtesi (infrared - IR) ve kamera teknolojileri listelenebilir. Bu bölümde sırasıyla SONAR ve IMU sensörlerine bakıp örnek projeler yapacağız. Özellikle IMU sensörü ile yaptığımız projede Arduino seri porttan gönderdiğimiz verileri bilgisayar tarafında Anaconda ve Python kullanarak PyGame kütüphanesi yardımıyla görselleştireceğiz.

3.1. HC-SR04 SONAR Sensör ile Mesafe Ölçümü

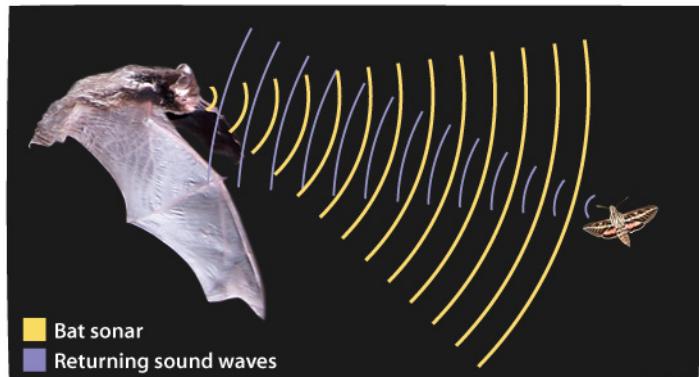
3.1.1. SONAR Sensör Teknolojisi

Bu teknolojiye ilham kaynağı olan canlıların en popülerleri yarasalardır. Karada yaşayan bir memeli olan yarasalar bilindiği gibi gözleriyle etraflarını göremezler. Diğer canlılardan farklı olarak yarasaların işitme sistemleri oldukça gelişmiştir. İnsan kulagini duyma frekansının (i.e., 20Hz-20kHz arası) üzerinde bir frekans değerine sahip sesler üretip çevreye gönderen yarasalar, etraftan yansiyarak kendilerine dönen (i.e., echo) sesleri kullanarak ortamın gerçek-zamanda (real-time) haritasını çıkarabilmekte ve böylece kendi konumlarını izafi olarak bulup navigationlarını sağlayabilmektedirler. Örnek olarak, Şekil 17'de sarı renkte dalgalar olarak gösterilen sesleri gönderen yarasa, kelebekten yansiyip geri dönen mor renkteki ses dalgalarını geri almakta böylece kendi beyinde kelebeğin nerede olduğunu sanki gözleriyle görmüş gibi oluşturabilmektedir.¹ Bizler insanlar olarak yarasanın çıkarmış olduğu bu sesleri duyamamaktayız fakat yarasanın avı olabilecek bazı böcekler (bazı kelebekler de dahil) bu sesleri duyarak yarasaların kendilerini avladığını anlayıp onlardan kaçabilmekte, hatta kendileri de (yne bizim duyamayacağımız frekanslarda) ses üretip yarasaları caydırabilmektedir.

Yarasa SONAR² teknolojisini kullanan canlılar arasında yalnız değildir. Denizde

¹<https://askabiologist.asu.edu/echolocation>, Echolocation, 17 Haziran 2022 tarihinde erişildi.

²SONAR sensörlerle超声波 sensor de denmektedir.



Şekil 17. Yarasanın kullandığı SONAR sistemi.

yaşayan memelilerden olan yunuslar, balinalar ve karada yaşayan bazı hayvanlar (özelikler bazı kuşlar) da bu yöntemle lokalizasyon yapabilmektedir.³ Şekil 18'de denizde yaşayan balinanın yaptığı SONAR konumlamayı ve bundan ilham alan insanoğlunun geliştirmiş olduğu denizaltını görebilirsiniz (sarı renkteki dalgalar gönderilen sesi, kırmızı renkteki dalgalar geri alınan sesi temsil etmektedir).⁴ Yarasanın karada üretip göndermiş olduğu sese benzer bir ses üreten balinalar bu sefer hava yerine suda sesi yollayıp cisimlerden yansiyan dalgaları geri alarak kendileriyle önlerindeki engeller arasındaki mesafeyi hesaplayabilmektedirler. Denizde SONAR kullanımına bir başka örnek olarak balık sürülerinin SONAR ile tespit edilmesi de verilebilir. Hayvanların aksine mühendisler SONAR teknolojisini hem karada (yani havada) hem denizde kullanmaktadır.⁵ SONAR sensörler ile mesafe ölçümünde dikkat edilmesi gereken en önemli nokta sesin ortamdan ortama değişen hızıdır.

3.1.2. Çalışma Prensibi

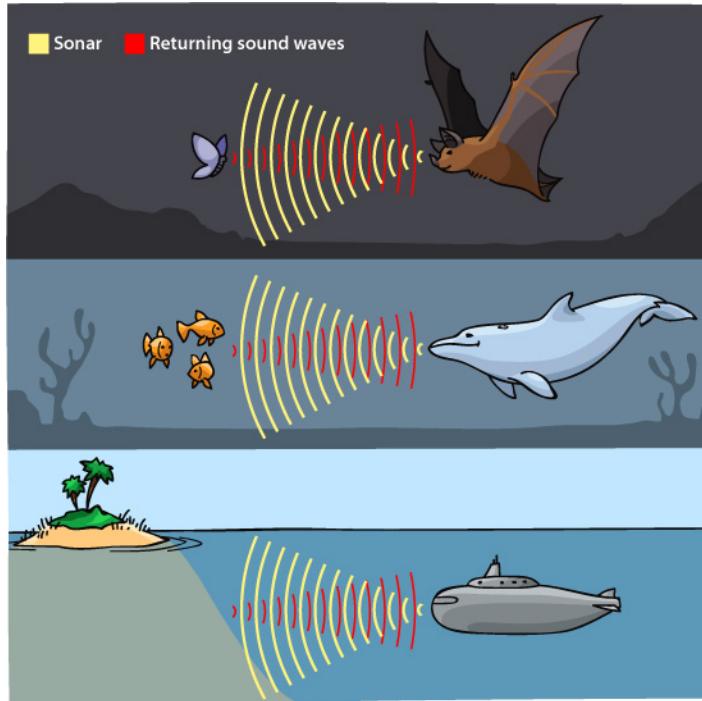
Şu ana kadar SONAR kullanan canlı veya cihazın insanoğlunun duyamayacağı bir ses ürettiğini ve engellere çarpan bu seslerin sesleri gönderen kaynak tarafından geri alınmasıyla mesafenin hesaplanabildiğini anladık. Peki bu mesafe hesabı tam olarak nasıl yapılmakta?

Fizikte bilindiği gibi hız ve zaman çarpımı yola eşittir. Başka bir deyişle, hız grafi-

³<https://a-z-animals.com/blog/top-10-animals-that-use-sonar-to-survive>, Hayatta kalmak için SONAR kullanan 10 canlı, 17 Haziran 2022 tarihinde erişildi

⁴<https://askabiologist.asu.edu/echolocation>, Echolocation, 17 Haziran 2022 tarihinde erişildi.

⁵Sesin iletilebildiği her ortamda SONAR teknolojisiyle mesafe hesaplanabilirken uzayda ses iletilemediği için SONAR teknolojisi kullanılmamaktadır.



Şekil 18. Yarasa, balina ve denizaltı SONAR sistemleri.

ğinin altındaki alandan (i.e., hızın integrali) konum hesaplanabilmektedir. Bu prensibi SONAR sensör ile mesafe ölçümüne uygulamamız için sesin gönderildiği andan tekrar kaynağa döndüğü ana kadar geçen süreyi bilmemiz gerekmektedir. Bu işi SONAR sensörün bağlanıp ölçümlerini okuduğu, mikrodenetleyici olarak vazife gören Arduino'nun yapısındaki zamanlayıcı (timer) veya sayıcı (counter) denilen kronometre ile yapabiliriz.

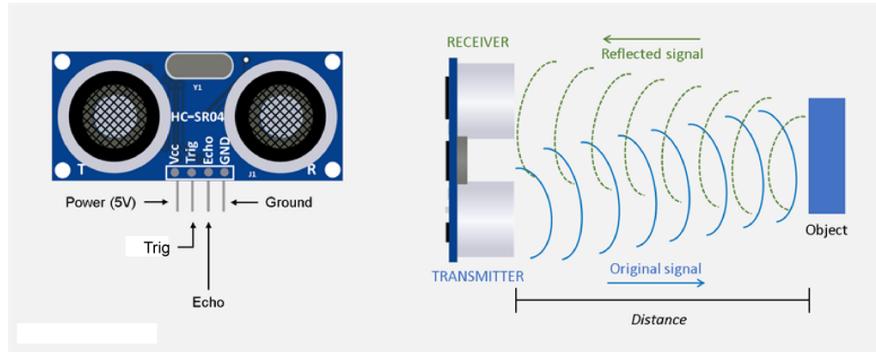
Bu çalışmada, SONAR sensörden gönderilen ses dalgaları yayılmak için ortam olarak havayı kullandığımızdan ses hızını $v \approx 340 \text{ m/s}$ kabul edeceğiz. Çalışmanın ilerleyen bölümlerinde (kod üzerinde) ayrıntılı olarak anlatılacak olan geçen süre hesabının yapıldığını ve Δt değişkeni ile temsil edildiğini varsayırsak o zaman Arduino mikrodenetleyicisi üzerinde mesafe hesabı aşağıdaki gibi yapılabilir.

$$x = 340\Delta t \quad (1)$$

Burada x metre cinsinden mesafeyi, Δt ise saniye olarak geçen zamanı temsil etmektedir. Bu projede kullanılan HC-SR04 ultrasonik sensörünün Trigger bacağından ses dalgasını gönderdiğimiz anda Arduino'nun zamanlayıcısında okunan değer t_0 , dalga engelden yansiyip sensörün Echo bacağına döndüğü anda Arduino'nun zamanlayıcısında okunan değer t_f olsun. Buradan

$$\Delta t = t_f - t_0 \quad (2)$$

olarak hesaplanabilir. HC-SR04 sensörü ile mesafe ölçümü Şekil 19'da gösterilmiştir.



Şekil 19. HC-SR04 SONAR sensör ile mesafe ölçümü.

3.1.3. HC-SR04 Ultrasonik Sensör

Bir HC-SR04 sensörü iki transdüberden oluşmaktadır. İlk elektrik sinyalini 40KHz frekansındaki ultrasonik sese dönüştürerek verici, ikincisi ise engellerden yansıyan ve sensöre geri dönen ultrasonik sesi elektrik sinyaline çevirerek alıcı işlevi görmektedir. Alıcı olan kısmı sesi aldığı anlarda ölçülen mesafeyle doğru orantıda genişliği sahip bir kare dalga üretmektedir.⁶ Bu sensör 3mm hassasiyetle 2cm'den 4m'ye kadar mesafeyi başarıyla ölçebilmektedir. Besleme gerilimi 5V olduğundan dolayı Arduino gibi üzerinde 5V çıkış pinleri olan mikrodenetleyicilerle uyumludur ve pratikte robotik uygulamalarda çok fazla tercih edilmektedir. Aşağıda Tablo 20a'de HC-SR04 sensörünün teknik özellikleri Şekil 20b'de ise pin dağılımı verilmiştir.

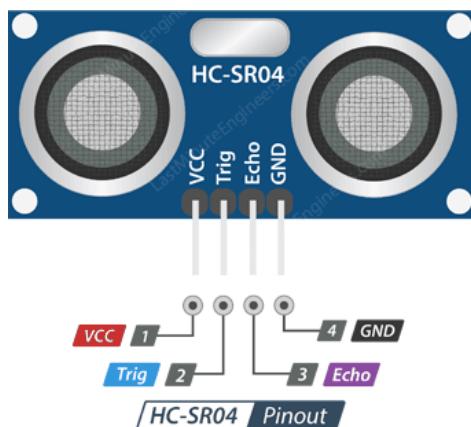
3.1.4. Arduino ile HC-SR04 SONAR Sensör Bağlantıları

Sensör ile Arduino'nun bağlantı şeması Şekil 21'de verilmiştir. Burada HC-SR04'ün Trigger ve Echo pinleri sırasıyla Arduino'nun dijital 9 ve 10 pinlerine bağlı gösterilmiştir.

⁶<https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/>, HC-SR04 Ultrasonik Sensör, 17 Haziran 2022 tarihinde erişildi.

Çalışma Voltajı	5V
Çalışma Akımı	15mA
Çalışma Frekansı	40KHz
Maksimum Ölçülebilen Mesafe	4m
Minimum Ölçülebilen Mesafe	2mm
Ölçme Hassasiyeti	3mm
Ölçme Açısı	15°
Boyutlar	45 x 20 x 15mm

(a) HC-SR04 Ultrasonik Sensör Teknik Özellikleri



(b) HC-SR04 SONAR sensör ile mesafe ölçübü.

Şekil 20. HC-SR04 ultrasonik sensör özellikleri ve pin dizilimi

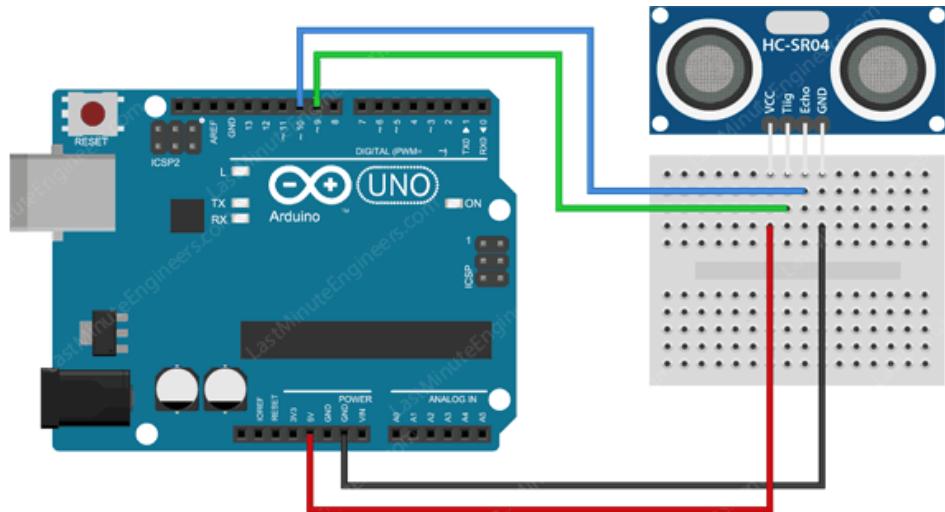
3.1.5. Mesafe Ölçümü Deneyi

Yukarıda verilen bilgiler ışığında SONAR sensörden gelen veriyi Arduino'ya okuyup mesafeyi cm cinsinden bularak seri port ekranına yazdırduğumuz kodu aşağıda bulabilirsiniz. Ayrıca Arduino'ya bir buzzer bağlayarak SONAR sensörden okunan mesafe değeri 5cm altına düştüğünde öten bir alarm sistemi eklenmiş, son olarak da alarm çalıştığında kırmızı bir LED'i, alarm sessiz olduğunda ise yeşil bir LED'i yakacak şekilde kod yazılmıştır.

```

1 #define redLedPin 6
2 #define greenLedPin 7
3 #define buzzerPin 8
4 #define trigPin 9
5 #define echoPin 10
6
7 void setup() {
8     Serial.begin(9600);

```



Sekil 21. Arduino – SONAR sensör (HC-SR04) bağlantıları.

```

9  pinMode(redLedPin, OUTPUT);
10 pinMode(greenLedPin, OUTPUT);
11 pinMode(buzzerPin, OUTPUT);
12 pinMode(trigPin, OUTPUT);
13 pinMode(echoPin, INPUT);
14 }
15
16 void loop() {
17     long duration;
18     int distance;
19     // Trigger the ultrasonic sensor
20     digitalWrite(trigPin, LOW);
21     delayMicroseconds(2);
22     digitalWrite(trigPin, HIGH);
23     delayMicroseconds(10);
24     digitalWrite(trigPin, LOW);
25     // Read echo and calculate distance in cm
26     duration = pulseIn(echoPin, HIGH); // returns time in microseconds
27     // The speed of sound in air is indeed ~340 m/s (34,000 cm/s)
28     // The time it takes for the sound to travel to the object and
29     // back is divided by 2 to get the one-way distance.
30     distance = duration * 0.034 / 2;
31     Serial.print("Distance: ");
32     Serial.print(distance);
33     Serial.println(" cm");

```

```

34 // Control LEDs and Buzzer
35 if (distance > 5) {
36     digitalWrite(redLedPin, LOW);
37     digitalWrite(greenLedPin, HIGH);
38     digitalWrite(buzzerPin, LOW); // Turn off buzzer
39 }
40 else {
41     digitalWrite(redLedPin, HIGH);
42     digitalWrite(greenLedPin, LOW);
43     digitalWrite(buzzerPin, HIGH); // Turn on buzzer
44 }
45 delay(50);
46 }

```

3.1.6. Fırçasız Motor - Pervane Hız Kontrol Deneyi

Yukarıda gerçeklediğimiz kodda SONAR sensör tarafından hesaplanan mesafe belli bir eşik değerinin altına düştüğünde LED yakıp/söndürüp buzzer’ı aktif/deaktif hale getirip tabiri caizse bir hırsız alarm devresi oluşturduk. Şimdi burada sensör tarafından okunan mesafeyle doğru orantılı olacak bir şekilde PWM sinyali üreterek fırçasız motor ve pervane sisteminin hızını kontrol edelim. İlgili kodu aşağıda bulabilirsiniz.

```

1 #include <Servo.h>
2
3 #define TRIG_PIN 9
4 #define ECHO_PIN 10
5
6 Servo motor;
7
8 void setup() {
9     Serial.begin(9600); // Start serial communication for debugging
10    pinMode(TRIG_PIN, OUTPUT);
11    pinMode(ECHO_PIN, INPUT);
12    motor.attach(3);
13 }
14
15 void loop() {

```

```

16 // Measure the distance using the ultrasonic sensor
17 long duration, distance;
18 // Trigger the sensor
19 digitalWrite(TRIG_PIN, LOW);
20 delayMicroseconds(2);
21 digitalWrite(TRIG_PIN, HIGH);
22 delayMicroseconds(10);
23 digitalWrite(TRIG_PIN, LOW);
24 // Read the echo pin
25 duration = pulseIn(ECHO_PIN, HIGH);
26 // Calculate the distance in cm (speed of sound is ~ 343 m/s)
27 distance = duration * 0.034 / 2;
28 // Map the distance to motor speed
29 uint16_t motorSpeed = map(distance, 10, 500, 900, 2100);
30 motor.writeMicroseconds(motorSpeed);
31 // Print distance for debugging
32 Serial.print("Distance: ");
33 Serial.print(distance);
34 Serial.print(" cm, Motor Speed: ");
35 Serial.println(motorSpeed);
36 delay(20);
37 }

```

3.2. MPU6050 Hareket Sensörü Uygulamaları

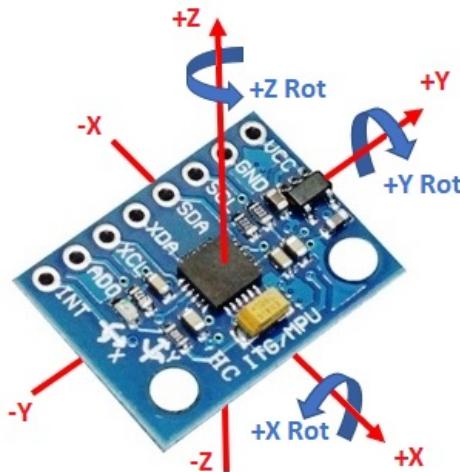
3.2.1. MPU6050 Hareket Sensörü

Burada hareket sensörü (motion sensor) diye isimlendirdiğimiz sensörün literatürdeki ismi Ataletsel Ölçüm Birimi (AÖB) olup İngilizce'si Inertial Measurement Unit (IMU) olarak geçmektedir. Bu açıdan MPU6050 ve veri aktarmak için bağlı olduğu Arduino beraber bir IMU oluşturmaktadır. Ataletsel sensörler iki tanedir: İvmemetre (accelerometer) ve jiroskop (gyroscope). İvmemetre m/s^2 veya g force cinsinden statik⁷ ve dinamik⁸ ivmeye, jiroskop ise rad/s veya $^o/s$ cinsinden açısal hızı ölçmektedir [2]. Ölçümler 3b uzayda gerçekleştiğinden dolayı IMU'lar x, y, z eksenlerinde değer üret-

⁷Dünyanın sabit yerçekimi kuvveti olarak kabul edilen $9.8m/s^2$.

⁸Hareket, şok veya titreme (vibrasyon).

mektedir. Dolayısıyla IMU 6-dof (6 degree of freedom yâni serbestlik derecesi) olarak geçmektedir [2]. Örnek bir MPU6050 sensörünü, üzerinde yer alan ortogonal eksenleri ve oluşturdukları Euler açlarını Şekil 22'de görebilirsiniz.



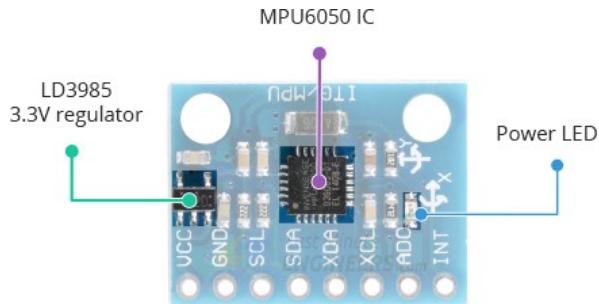
Şekil 22. MPU6050 hareket sensörü üzerinde x, y, z eksenleri.

3.2.2. MPU6050 Modülü Donanımsal İnceleme

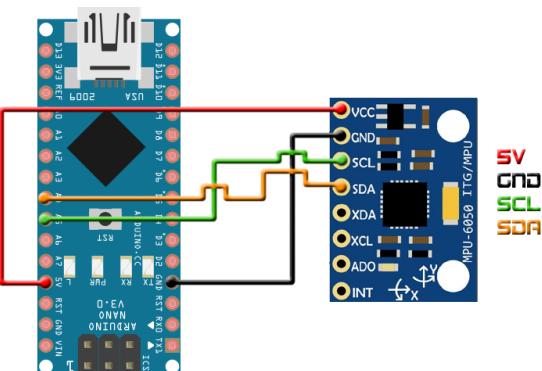
MPU6050 modülünün merkezinde 3 eksenli ivmemetreyi ve jiroskopu ve bu atatelsel sensörlerden yakalanan ham verileri işleyen bir dijital hareket işlemcisini (digital motion processor - DMP) 4mm x 4mm'lik minicik bünyesinde toplayan düşük güç ve mâliyetli MPU6050 entegresi yer almaktadır [2]. Şekil 23'te modüle SMD (surface mount device) biçimde lehimlendiği görülen bu çekirdeğin hemen yanında yer alan LD3985 3.3V regulatör sayesinde MPU6050 modülü Vcc pininden 5V olarak beslenebilmektedir. Kullanıldığı uygulamalarda çalışırken 3.6mA'den daha az akım tüketen MPU6050, rölatif (idle) hâlinde sadece $5\mu A$ çekmektedir. Bu düşük güç tüketimi sayseinde MPU6050 birçok uygulamada tercih edilmektedir. Ayrıca Şekil 23'te görülen LED sayesinde modülün çalışıp çalışmadığı (doğru bağlantıların yapılp yapılmadığı) gözle kolayca anlaşılabilmektedir.

3.2.3. MPU6050-Arduino Bağlantıları

Şekil 24'de MPU6050 hareket sensörüyle Arduino arasındaki bağlantıları görebilirsiniz.



Şekil 23. MPU6050 modülü.



Şekil 24. MPU6050-Arduino bağlantıları.

Şekil 25'de MPU6050 hareket sensörü ve Arduino ile breadboard üzerine kurulan devreyi görebilirsiniz.

3.2.4. MPU6050 Hareket Sensörü ile Euler Açıları Hesabı

İnternette ve Arduino kütüphane sisteminde birçok MPU6050 Euler açıları tahoma kodu bulabilirsiniz. Bunlar arasında bizim için en uygun olanı anlaşılması kolaylığı yönünden **tockn**⁹ kütüphanesidir. Yukarıda Şekil 24 ve Şekil 25'te görülen bağlantıları yaptıktan sonra **tockn** kütüphanesini yüklemek için Arduino IDE'de yukarı tarafta yer alan Sketch tab'de

Sketch → Include Library → Manage Libraries

yolunu takip ediniz. Şekil 26'da bu işi nasıl yapabileceğinizi görebilirsiniz.

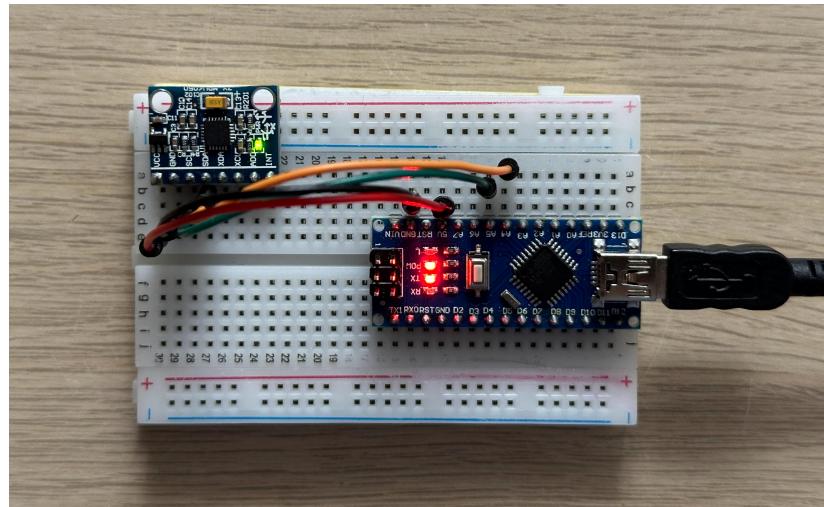
Hemen ardından gelen ekranda **tockn** kütüphanesini Şekil 27'de gösterildiği gibi

INSTALL

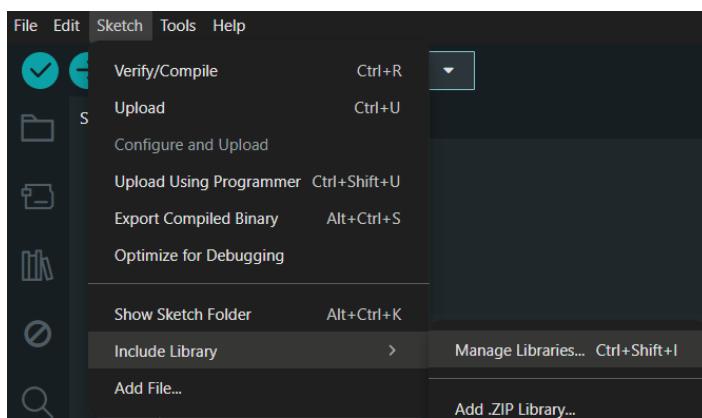
ikonuna tıklayarak yükleyiniz.

Yüklemeler tamamlandıktan sonra bilgisayarda **Belgeler** klasörüne yüklenen Euler

⁹https://github.com/tockn/MPU6050_tockn



Şekil 25. MPU6050-Arduino devresi (breadboard).



Şekil 26. Arduino IDE'de kütüphane ekleme.

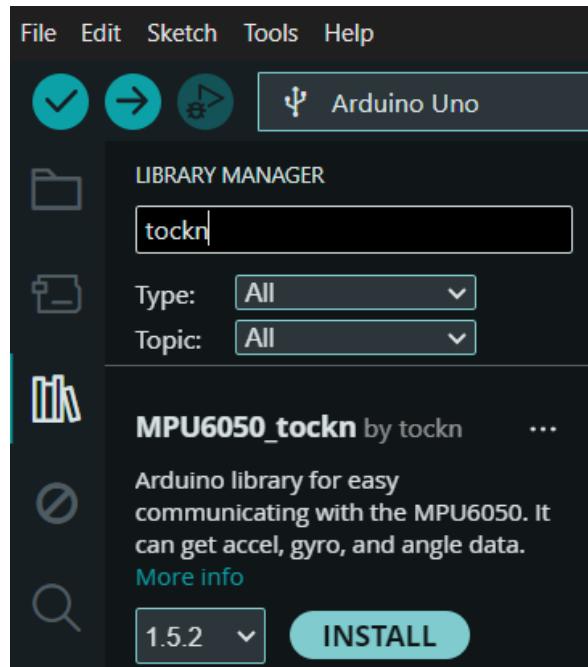
açılıları kodunu hesaplayan **tockn** kütüphanesi örneği **GetAngle.ino** kodunu Şekil 28 gibi açarak test ediniz.¹⁰ **Not:** Şekil 24 ve Şekil 25'te gösterilen bağlantıları yaptıgınızda emin olunuz.

```

1 #include <MPU6050_toockn.h>
2 #include <Wire.h>
3
4 MPU6050 mpu6050(Wire);
5
6 void setup() {
7     Serial.begin(9600);
8     Wire.begin();
9     mpu6050.begin();
10    mpu6050.calcGyroOffsets(true);

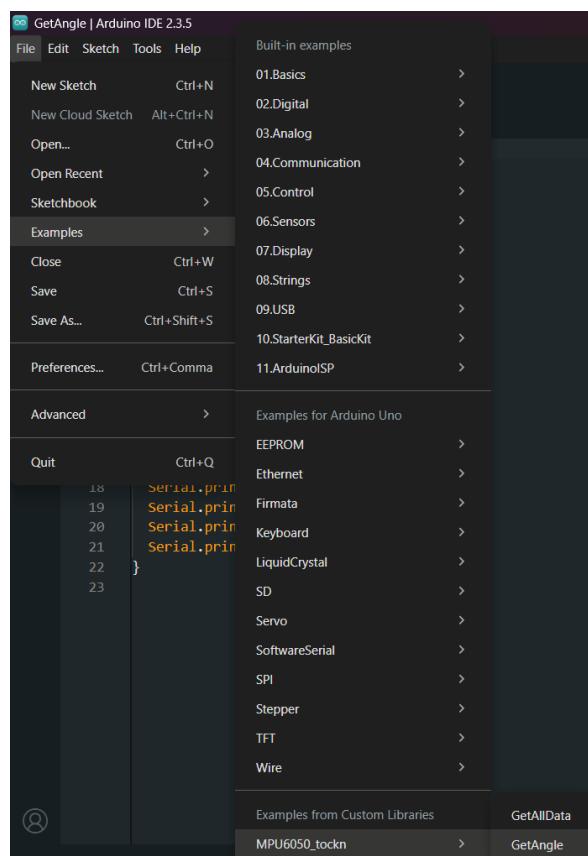
```

¹⁰İlgili videoyu <https://www.youtube.com/watch?v=O15onAIy4no> bağlantısından izleyebilirsiniz.



Şekil 27. Arduino IDE'de **tockn** kütüphanesini kurma.

```
11  }
12
13 void loop() {
14     mpu6050.update();
15     Serial.print("angleX : ");
16     Serial.print(mpu6050.getAngleX());
17     Serial.print("\tangleY : ");
18     Serial.print(mpu6050.getAngleY());
19     Serial.print("\tangleZ : ");
20     Serial.println(mpu6050.getAngleZ());
21 }
```



Sekil 28. Arduino IDE'de **tockn** kütüphanesi **GetAngle.ino** örneğini açma.