# 5. Hafta – Python Programlama ile Robotu Hareket Ettirme

## Haftanın Amacı:

Bu haftanın amacı öğrencilere robot setini ve robot setini programlamak için kullanılacak olan Python ortamını tanıtmaktır. Öğrenciler robotun belirli bir mesafe boyunca hareket etmesi, viraj alma, kendi ekseninde dönme, bir teker etrafında dönme gibi manevraları yapması ve motorlar ile keskin hareketler gerçekleştirmesi beklenmektedir.

## Haftanın Kazanımları:

* Öğrenciler orta ve büyük motoru tanır.
* Robotu programlamak için Python ortamını kullanır.
* Robotun keskin manevra ve hareketler yapabilmesi için ilgili komutlar ile günlük yaşam verisini birleştirir.
* Problem çözümü için hareket komutlarını bir arada kullanır.

## Kullanılacak Malzemeler:

Robot seti, bilgisayar ve çalışma alanları.

## GÖZLE VE UYGULA

|  |
| --- |
| ***Rehber Öğretmen İçeriği –*** ***Python programlamaya diline giriş*** |
| *Bu hafta metin tabanlı programlama dillerinden olan Python programla dilini kullanarak örnekler yapmaya başlanacaktır. Bunun için Spike Prime Yazılımı arayüzünden “Yeni Proje” seçeneği ile ilk örnek oluşturulur.*  metin içeren bir resim  Açıklama otomatik olarak oluşturuldu  *Gelen ekrandan oluşturulacak projenin adı verilir. Örneğin projenin adı “ornek1” olarak giriş yapılır ve PYTHON seçilerek programlama arayüzüne geçilir.*    *Yeni proje ile metinsel programlama arayüzü açılır ve arayüzde neler olduğu gösterilir. Blok tabanlı programlama arayüzü ile benzerlikler ve farklılıklar vurgulanır. Daha sonra yazılımının bölümleri tanıtılır.*    *Arayüzün sağ tarafından “Bilgi Bankası” bölümünden bu ders boyunca da işleyeceğimiz birçok konu hakkında özet bilgilere erişilebileceği görülür. Arayüzün programlama alanına kodlar yazılarak sağ altındaki oynat düğmesi (sarı düğme) ile yazdığımız programları Hub’a gönderilerek çalıştırılabilir. Ekrandaki ilk kodların çalıştırılması durumunda Hub’ın ekranında gülen yüz ifadesi görülür.* |

### Gözle: Sürüş Tabanında Direksiyon Hareketi

Bu etkinlikte amaç robotun hareketini sağlamaktır. Robotun hareketini sağlamak için öncelikle robotun tanıtılması gerekmektedir. Bu etkinlikte Sürüş Modeli 2’nin 26. adımına kadar olan inşası kullanılacaktır. Sürüş modeli bu kitapta sürüş tabanı olarak adlandırılacağından robotun ismi *surus\_tabani* olarak belirlenmiştir. Sürüş tabanının tanımlanması için bir *MotorPair* nesnesi oluşturulmalıdır. *MotorPair* nesnesi oluşturulurken iki parametre belirlenmelidir. İlk parametre sol motorun bağlı olduğu portu, ikinci parametre ise sağ motorun bağlı olduğu portu belirlemek için kullanılır. Sürüş tabanının sol ve sağ motorları sırasıyla C ve D portlarına bağlanmıştır. Sürüş tabanı aşağıdaki kod kullanılarak oluşturulur.

surus\_tabani=MotorPair("C","D")

Sürüş tabanı oluşturulduktan sonra ileri doğru doğrusal hareketi sağlamak için *move* metodu kullanılabilir. Move metodunun parametreleri detaylıca incelenmeden önce bir örnek vermek yerinde olacaktır. Aşağıdaki kod ile sürüş tabanı oluşturulmuş ve ileriye doğru 40 hızında 20 cm hareket etmesi sağlanmıştır. Rehber öğretmen kodu tahtaya yansıtır, programı çalıştırır ve öğrencilere kodu ve kodun görevini açıklar.

surus\_tabani=MotorPair("C","D")

surus\_tabani.move(20,"cm",0,40)

Ardından aşağıdaki kodu tahtaya yansıtır ve çalıştırır. Bu kodun görevinin sürüş tabanını 30 cm ileriye doğru hareket ettirmek olduğunu belirtir. Rehber öğretmen her iki kodu da öğrencilerin yazıp çalıştırmasını sağlar.

surus\_tabani=MotorPair("C","D")

surus\_tabani.move(30,"cm",0,40)

### Uygula: Direksiyon Hareketini Keşfediyorum

Rehber öğretmen öğrencilere aşağıdaki soruyu sorar:

Move metodunun kullanımını keşfetmek için bir önceki etkinlikte yazılan *move* metodu üzerinde değişiklikler yaparak robotu ileri, geri, sağa ve sola hareket ettirmeye çalışınız.

Rehber öğretmen öğrencilere yeteri kadar süre verir. Bu sürenin sonunda bazı gruplardan soru ile ilgili bulgularını sınıf önünde açıklamalarını ister.

### Gözle: Direksiyon Hareketinin Detaylarını Öğreniyorum

Move metodu 4 parametre almaktadır:

move(hareket miktarı, hareket birimi, yön, hız)

Hareket miktarı parametresi sürüş tabanının ne kadar ve hangi doğrultuda (pozitif değerler için ileri ve negatif değerler için geri) hareket edeceğini belirler. Hareket miktarında referans noktası olarak sürüş tabanının iki tekerleğinin orta noktası esas alınır. Yani iki tekerleği birleştiren hayali bir doğru çizildiğinde bu doğrunun orta noktası belirlenen hareket miktarı kadar yer değiştirecektir. Hareket birimi parametresi ise hareket miktarın birimini belirlemek için kullanılır. Hareket birimi olarak *“cm”, “in” (inç), “rotations” (dönüş), “degrees” (derece), “seconds” (saniye)* değerleri girilebilir. Örneğin hareket miktarı olarak 10 ve hareket birimi olarak “cm” girilmişse 10 cm’lik ileri doğru bir hareket yapılacağı belirtilmiş olur. Hareket miktarı olarak -10 girildiğinde aynı hareket geri doğru yapılır. Başka bir örnek vermek gerekirse hareket miktarı olarak 180 ve hareket birimi olarak *“degrees”* girilmişse tekerlekler toplamda 180 derecelik bir hareket meydana getirecektir. Hareket birimi çift tırnak içerisinde yazılmıştır. Eğer bu değer çift tırnak içerisinde yazılmazsa hata mesajı alınır.

|  |
| --- |
| Not |
| Hareket birimi çift tırnak yerine tek tırnak işareti içine de alınabilir. Bu şekilde de kod çalışacaktır. Python’da tek tırnak veya çift tırnak içerisine alınan ifadelere *metin (String)* denilir ve tek tırnak veya çift tırnak arasına alma arasında bir fark bulunmamaktadır. Bu kitap boyunca çift tırnak işareti kullanılmıştır. İstenirse tek tırnak işareti de kullanabilir. |

Yön parametresi 0 verilirse robot düz ilerler, negatif değerler için sola ve pozitif değerler için sağa döner. Yön değerine -100 ile 100 arasında değerler verilebilir. Yönün mutlak olarak değeri arttıkça dönüş keskinliği artacaktır. -100 ve 100 değerleri için robot dönüşü iyice keskinleşir ve robot kendi merkezinde (-100 için sola ve 100 için sağa) dönecektir. Son parametre olan hız ile robotun hareket hızı belirtilir. Mutlak değer olarak 0’dan 100’e doğru gidildikçe motorun hızı artar. Pozitif değerler için robot ileriye ve negatif değerler için robot geriye doğru hareket edecektir. 100 ile motor ileri doğru en yüksek hızda ve -100 ile geriye doğru en yüksek hızda hareket eder.

Rehber öğretmen öğrencilere *move* metodunun parametrelerini örnekleyerek anlatır. Her bir parametre için ayrı ayrı örnek verilir. Fakat aşağıdaki örnekler özellikle vurgulanmalıdır:

i) move(36, “cm”, -100, 50) ile move(36, “cm”, -50, 50) arasındaki fark: -100 değeri için kendi ekseni etrafında dönüş gerçekleştirirken, geriye kalan negatif değerlerde ise viraj döner şekilde hareket ettiğini vurgulanır.

ii) move(360,”degrees”,-50,50) komutu ile robot 360 derece dönmemektedir. Bunun yerine tekerler 360 derece yani tam tur dönmektedir.

iii) move(20,”cm”,0,50), move(-20,”cm”,0,50), move(20,”cm”,0,-50) ve move(-20,”cm”,0,-50) komutları kullanılarak hareket miktarı ile hızın işaretleri arasındaki ilişki anlatılır: geri veya ileri hareket için sadece hareket miktarının veya sadece hızın işaretinin kullanılması yeterlidir. Aynı anda hareket miktarı ve hızın işareti negatif yapıldığında robot ileri doğru hareket edecektir.

### Uygula: İlerleyip Aynı Yere Gelen Robot

Öğrencilerden

1. robotun 30 cm ileri gitmesini,
2. robotun geri dönmesini,
3. geri gelerek başlangıç noktasına ulaşmasını
4. ve yüzünü ilk başlangıç haline dönmesini sağlayan programı yazmaları istenir.

Bu görev için aşağıdaki kod kullanılabilir. Burada robotun 180 derece dönmesini sağlamak için 18 cm’lik kendi ekseni etrafında sola doğru bir dönüş kullanılmıştır. Öğrenciler farklı şekillerde ve değerler kullanarak dönüş işlemlerini gerçekleştirebilir. Rehber öğretmen öğrencilerden birebir aşağıdaki kodu yazmasını beklememelidir. Öğrencilerin kendi yollarından çözüme ulaşması önemlidir.

surus\_tabani=MotorPair("C","D")

surus\_tabani.move(30,"cm",0,50)

surus\_tabani.move(18,"cm",-100,50)

surus\_tabani.move(30,"cm",0,50)

surus\_tabani.move(18,"cm",-100,50)

### Uygula: Tekerin Yarıçapını Hesaplıyorum

Bu etkinlikte öğrencilerden tekerlerin bir tur döndürüldüğünde robotun aldığı mesafeyi cm cinsinden bulmaları istenir. Ardından öğrencilerden tekerin yarıçapını hesaplamaları istenir. Spike Prime ile gelen tekerler standart küçük teker olarak adlandırılır ve bu tekerin çapı 5.6 cm’dir. Tam tur teker dönüşünde robot 17.6 cm yol alır. Öğrencilerin kendilerinin hesaplaması yapması beklenir ve hesaplamalarının ardından bu değer onlara söylenir.

### Uygula: 360 Derece Dönen Robot

Öğrencilere iki tur kendi ekseni etrafında dönen robotun yaklaşık olarak kendi ekseni etrafında 360 derece döndüğü söylenir ve aşağıdaki kodu kullanarak bu durumu gözlemeleri beklenir. Öğrencilerden bu durumun neden böyle olduğunu açıklamaları istenir.

surus\_tabani=MotorPair("C","D")

surus\_tabani.move(2,"rotations",-100,50)

Sol teker ve sağ teker arasında yaklaşık olarak 11.2 cm mesafe (teker çapının iki katı) bulunmaktadır. Robot kendi ekseni etrafında döndüğünde çapı yaklaşık olarak 11.2 cm olan dairesel bir hareket yapar. Bu değer tekerlerin 2 tam tur (35.2 cm) dönmesi ile elde edilir.



*Resim 5.1 Kendi Ekseninde Dönme*

### Gözle: Sürüş Tabanında Palet Hareketi

Direksiyon hareketinde dönülecek yön ve dönüş miktarı bildirildikten sonra tekerlere verilecek hız büyüklüğünü robotun algoritması belirler. Direksiyon Palet hareketi direksiyon hareketinden farklı olarak sol ve sağ motorlara verilecek hız miktarını programcının belirlemesine imkân sağlar. Palet hareketi için *move\_tank* metodu kullanılır. Bu metotta dört parametre bulunmaktadır:

move\_tank(hareket miktarı, hareket birimi, sol teker hızı, sağ teker hızı)

İlk iki parametre move metodu ile aynıdır. Hareket miktarı iki tekerleğin orta noktasının yaptığı toplam yer değiştirmeyi gösterir. Hareket miktarı negatif veya pozitif değerler alabilir. Sol ve sağ teker hızlarının pozitif olduğu varsayılırsa; hareket miktarının pozitif değerlerinde ileri ve negatif değerlerde ise geriye doğru hareket sağlanır. Hareket birimi ise hareket miktarının cinsini belirler. Hareket birimi olarak *“cm”, “in” (inç), “rotations” (dönüş), “degrees” (derece), “seconds” (saniye*) değerleri girilebilir. Sol teker hızı ve sağ teker hızı parametreleri sol ve sağ tekerlerin hızını ayrı ayrı belirlemek için kullanılır. Bu parametreler -100 ile 100 arasında değer alır. Negatif hızlar geriye doğru ve pozitif hızlar ileri doğru hareket için kullanılır.

### Uygula: Sürüş Tabanında Palet Hareketi

Bu etkinlikte amaç öğrencilerin palet hareketinin sağladığı manevra çeşitliliğini görmelerini sağlamaktır. Bu iş için öğrencilerden aşağıdaki üç farklı hareket için kod yazmaları istenir:

i) Virajı alır şekilde dönme (surus\_tabani.move\_tank(20,"cm",40,80)),

ii) Kendi ekseninde dönme (surus\_tabani.move\_tank(20,"cm",40,-40)),

(iii) Bir tekerlek (sol veya sağ) etrafında dönme (surus\_tabani.move\_tank(20,"cm",0,80)).

### Gözle: Robotun Özelliklerini Ayarlıyorum

Sürüş tabanının ön tanımlı hızı, teker çevre uzunluğu ve durma şekli gibi özellikler kod yardımıyla belirlenebilir. Ön tanımlı hızı belirlemek için aşağıdaki komut kullanılabilir.

surus\_tabani.set\_default\_speed(50)

Bu komut ile tekerlerin ön tanımlı hızı 50 olarak belirlenir. Ön tanımlı hız belirlendikten sonra, move ve move\_tank fonksiyonu içerisinde tekerlerin hızını belirtilmezse bu ön tanımlı hız kullanılır. Örneğin aşağıdaki komut ile sürüş tabanı 50 hızıyla 20 cm doğrusal ileri yönlü bir hareket gerçekleştirir.

surus\_tabani.move(20,"cm",0)

Yukarıdaki komutta hız miktarı belirtilmediği için ön tanımlı hız değeri olan 50 kullanılmıştır. Hız değeri belirtmemek yerine hız için *None* değeri de kullanılabilir. Örneğin aşağıdaki komut ile sürüş tabanı yine 50 hızıyla 20 cm doğrusal ileri yönlü bir hareket gerçekleştirir.

surus\_tabani.move(20,"cm",0,None)

*Move\_tank* komutunda ön tanımlı hızı kullanması istenen tekere değer olarak *None* girilir. Aşağıdaki kodda sol teker için *None* değeri girildiğinden sol teker ön tanımlı değer olan 50’yi kullanacaktır ve sürüş tabanı 20 cm doğrusal ileri yönlü bir hareket yapacaktır. Aynı şekilde sağ teker için de *None* değeri girilirse, sağ teker de ön tanımlı hız değerini kullanır.

surus\_tabani.set\_default\_speed(50)

surus\_tabani.move\_tank(20,"cm",None,50)

Rehber öğretmen öğrencilere farklı *move ve move\_tank* komutları ile ön tanımlı hızı örneklendirir. *None* değerinin kullanımını ve hız değerini girmeme durumlarını açıklar.

Sürüş tabanının bir diğer özelliği ise kullanılan tekerin çevre uzunluğudur. Spike Prime seti ile standart küçük teker gelmektedir. Bu tekerin yarı çapı 2.8 cm’dir. Bu tekerin çevresi ise 17.6 cm yapar. Spike Prime eklenti setinde standart büyük teker bulunmaktadır. Bu tekerin yarıçapı 4.4 cm’dir. Bu tekerin çevresi ise 27.6 cm yapmaktadır. Sürüş tabanında ön tanımlı olarak küçük tekerlerin değeri kullanılmaktadır. Tekerlerin boyutu değiştiğinde *move ve move\_tank* metotlarında verilen hareket miktarları doğru sonuç vermez. Bu yüzden sürüş tabanının tekerleri değiştirildiğinde bunun robota bildirilmesi gerekir. Bu iş için *set\_motor\_rotation* metodu kullanılır. Örneğin büyük tekerlerin sürüş tabanına takılması durumunda aşağıdaki komut kullanılabilir.

surus\_tabani.set\_motor\_rotation(27.6,"cm")

Rehber öğretmen eklenti setindeki büyük tekerleri sürüş tabanına takar. Teker çevre uzunluğu özelliğini değiştirmeden aşağıdaki komutu çalıştırır.

surus\_tabani.move(30,"cm",0,50)

Daha sonra sürüş tabanının yer değiştirme miktarını ölçer. Çıkan sonucun 30 cm’den farklı olduğunu öğrencilere söyler. Bunun sebebinin ne olabileceğini onlara sorarak tartıştırır. Tartışma sonrasında tekerin çevre değerini 27.6 cm yaparak aynı kodu yeniden çalıştırır. Sonuçları öğrencilerle paylaşır ve sebebini öğrencilere anlatır.

Sürüş tabanında bahsedilmesi gereken son özellik durma şeklidir. Sürüş tabanında üç farklı durma şekli bulunmaktadır. Bunlar *“hold”, “brake” ve coast’ur*. *Hold* seçildiğinde tekerler ani bir şekilde bulunduğu yerde durur ve dışarıdan uygulanan kuvvet (yer çekimi, el ile itme gibi) ile tekerleri hareket ettirmek mümkün olmaz. Başka bir değişle sürüş tabanı bulunduğu yere sabitlenir. *Brake* seçildiğinde tekerler hızlıca durur fakat durma sonrasında dışarıdan uygulanan bir kuvvet ile hareket edebilir. *Coast* seçildiğinde tekerler görece yavaş bir şekilde durur ve dışarıdan uygulanan bir kuvvet ile hareket edebilir. Aşağıdaki komutlar kullanılarak istenilen durma şekli ayarlanabilir:

surus\_tabani.set\_stop\_action("hold")

surus\_tabani.set\_stop\_action("brake")

surus\_tabani.set\_stop\_action("coast")

Rehber öğretmen aşağıdaki komutu farklı durma şekilleri belirlendikten sonra çalıştırır ve her bir şekil için durma mesafesi ve durma sonrası hareket edip etmeme durumlarını öğrencilere gösterir.

surus\_tabani.move(50,"cm",0,80)

### Gözle: Motoru Belirli Bir Açıya Getirme

Sürüş tabanı gibi robot araba benzeri cihazların hareket ettirilmesi için motor çiftine yönelik komutlara ihtiyaç duyulur. Fakat bazı durumlarda motor çiftinden ziyade motorların ayrı ayrı hareketinin sağlanması gerekebilir. Bu durumda öncelikle aşağıdaki komut kullanılarak Motor cinsinden bir nesnenin oluşturulması gerekmektedir.

motor=Motor("E")

Sürüş tabanında bulunan büyük motor E portuna takılı olduğu için Motor sınıfından nesne oluştururken “E” parametresi kullanılmıştır. Bu komut yazıldıktan sonra sürüş tabanında bulunan büyük motora Motor sınıfının metotlarını kullanarak çeşitli komutlar verilip özellikler atanabilir. Motoru istenen bir açıya getirmek için *run\_to\_position* isimli metot kullanılabilir. Bu metot ile motorun pozisyonu bulunduğu açı değerinden hedef açı değerine doğru hareket ettirilir.

run\_to\_position(hedef motor açısı, yön, hız)

Hedef motor açısı parametresine derece cinsinden motorun ulaşması istendiği açı değeri girilir. Bu değerler 0 ile 359 arasında değişir. Yön parametresi ile istenen açıya hangi yönden gidilebileceği belirtilir. Yön üç farklı değer alabilir. Bunlar *“clockwise”, “counterclockwise” ve “shortest path”* değerleridir. *Clockwise* değeri Resim 5.2’de görüldüğü gibi motoru saat yönünde çevirir. *Clockwise* ile sürüş tabanının önünde bulunan hareketli kol yukarı doğru hareket eder. *Counterclockwise* saat yönünün tersine hareket sağlar. Sürüş tabanındaki hareketli kol aşağı doğru hareket edecektir. *Shortest path* ise bulunan açı ile hedef açı arasındaki en kısa yolu seçer. Örneğin motor 300 derecede duruyor olsun. 340 dereceye gitmek için motor saat yönünde (clockwise) hareketi tercih edecektir. Son olarak hız parametresi 0 ile 100 arasında değişen değerler alabilir. Buraya değer girilmemişse *set\_default\_speed* metodu ile belirtilen ön tanımlı değer kullanılır.



*Resim 5.2 Motor Dönüş Yönü ve Sıfır Noktası*

Resim 5.2’de gösterildiği gibi motorlar üzerinde sıfır dereceyi gösteren bir ibare bulunmaktadır (0 Noktası). Motorun işaretli kısmı bu noktaya geldiğinde motor sıfır derecede olacaktır. Bunun dışında motorun açısını bulmak için kod yazılan ekranın üst kısmındaki panelde bulunan motor açı değeri kullanılabilir. Resim 5.3’de örnek açı değeri gösterilmiştir.



*Resim 5.3. Motor Açısı*

Rehber öğretmen sürüş tabanına bağlı olmayan bir büyük motoru hub’a bağlayarak *run\_to\_position* metodunun farklı kullanımlarını onlara gösterir. Bunu yaparken farklı açı, yön ve hız değerlerini kullanır. Hız değeri negatif verildiğinde motorun verilen yönün tersine dönmediğini ve mutlak değer cinsinden hareket ettiğini vurgular.

### Uygula: İşaretçi Direk Eşleştirmesi

Rehber öğretmen öğrencilerden Resim 5.4’de gösterildiği gibi bir düzenek kurmalarını ister. Düzenek kurulduktan sonra geniş motor etrafındaki üçlü direği her bir öğrenci grubu için geniş motor etrafında rasgele bölgelere yerleştirir. Öğrencilerden, geniş motora bağlı işaretçinin sadece direklerin karşısında geldiğinde duracak şekilde, robotun kodlanmasını ister.



*Resim 5.4 Etkinlik Düzeneği*

### Gözle: Motoru Belirlenen Başlangıç Noktasına Göre İstenen Bir Açıya Getirme

*Run\_to\_position* metodu ile motor 0-359 arasındaki mutlak açı değerlerine getirilebilir. Fakat bazı durumlarda motor mutlak açı değerleri yerine göreceli bir açı değerine getirilmek istenebilir. Göreceli bir açı değerinden bahsedebilmek için öncelikle başlangıç değerinin belirlenmesi gerekmektedir. Robot ilk çalıştırıldığında motorun mutlak açı değeri ne olursa olsun, bulunan yer için göreceli açı değeri 0 olarak alınır. Bunun yanında istendiği zaman başlangıç açı değeri set\_degrees\_counted metodu ile belirlenebilir. Bu metodun kullanımı aşağıda gösterilmiştir.

set\_degrees\_counted(başlangıç açı değeri)

Bu metot ile motor herhangi bir mutlak açı değerinde duruyor iken (örneğin 180 derecede durduğunu düşünelim) set\_degrees\_counted(0) komutu ile bulunan yerin göreceli açı değerinin 0 olması sağlanır. Mutlak açı değerinde herhangi bir değişiklik olmamıştır (hala mutlak açı değeri 180 derecedir). Fakat bulunan yerin göreceli değeri artık 0 olarak kabul edilir. Bu metoda 0 değeri girmek zorunlu değildir. İstenen tam sayı değeri verilebilir. Verilen tam sayı değeri bulunan pozisyonun göreceli açı değeri olarak kabul edilecektir. Göreceli açı değerine göre belirli bir açıya hareket etmek için run\_to\_degrees\_counted metodu kullanılabilir. Bu metodun kullanımı aşağıdaki şekildedir.

run\_to\_degrees\_counted(hedef açı, hız)

Hedef açı değeri sayesinde göreceli açı değeri hedef değere ulaşana kadar motor hareket eder. Hedef açı bütün tam sayı değerlerini alabilir. Hız ise istenen hedef açıya hangi hız ile gidileceğini belirtir. Hız 0 ile 100 arasında bütün değerleri alabilir. Örnek vermek gerekirse bulunan pozisyonun açı değeri 50 olarak belirlenmişse hedef açı 100 girildiğinde, göreceli açı değerini 100 yapabilmek için motor saat yönünde 50 derece dönecektir. Bunun ardından ikinci bir komut ile hedef açı 80 olarak belirlenirse motor göreceli açı değerini 80 yapabilmek için saat yönünün tersine 20 dere dönecektir. Göreceli açı değeri Python’da degrees\_counted olarak isimlendirilir.

|  |
| --- |
| Not |
| Python’da *degrees\_counted* gibi bir üye değişkenin değerinin değiştirmek için kullanılacak metodun ismini belirlerken genellikle değişken isminin önüne set\_ ifadesi getirilir: *set\_degrees\_counted*. Bu değişkenin değerini öğrenmek için kullanılacak metodun ismini belirlerken değişken ismi önüne *get\_* ifadesi getirilir: *get\_degrees\_counted*. Bu durum bir zorunluluk olmamakla beraber bahsi geçen kullanım oldukça yaygındır. |

Rehber öğretmen öncelikle motorun mutlak açı değerini eli ile 0 derece yapar. Ardından aşağıdaki kodu çalıştırır ve motorun bulunduğu yerin göreceli açı değerini 50 derece olarak belirlediğini ve 40 derece saat yönünde döndüğünü anlatır. Motorun göreceli açı değeri 90 iken mutlak açı değeri 40 olacaktır.

motor=Motor("E")

motor.set\_degrees\_counted(50)

motor.run\_to\_degrees\_counted(90)

Rehber öğretmen aşağıdaki kodu çalıştırır ve motorun önce saat yönünde 90 derece hareket ettiğini daha sonra 180 dereceye ilerleyebilmek için saat yönünde doksan derece daha hareket ettiğini öğrencilere anlatır.

motor=Motor("E")

motor.set\_degrees\_counted(0)

motor.run\_to\_degrees\_counted(90)

motor.run\_to\_degrees\_counted(180)

Rehber öğretmen aşağıdaki kodu çalıştırır ve motorun önce saat yönünde 90 derece hareket ettiğini ardından *degrees\_counted* değerini -90’a getirebilmek için saat yönünün tersine 180 derece döndüğünü anlatır.

motor=Motor("E")

motor.set\_degrees\_counted(0)

motor.run\_to\_degrees\_counted(90)

motor.run\_to\_degrees\_counted(-90)

Rehber öğretmen aşağıdaki kodu çalıştırır ve öğrencilere *degrees\_counted* değerinin 720 olabilmesi için motorun yaklaşık olarak iki tur döndüğünü açıklar.

motor=Motor("E")

motor.set\_degrees\_counted(0)

motor.run\_to\_degrees\_counted(720)

### Uygula: Üç Turda İşaretçi Direk Eşleştirmesi

Bu etkinlikte, bir önceki etkinlikte yapılan işaretçi direk eşleştirmesinin bir benzeri yapılacaktır. Fakat bu kez işaretçi üç tur atacaktır. Her bir turda da işaretçinin sadece direkler karşısında durması gerekmektedir. Rehber öğretmen öğrencilerden gerekli kodu yazmalarını ister.

### Gözle: Belirli Bir Derece/Tur/Saniye Hareket Etme

Bazı durumlarda motorun mutlak veya göreceli istenen açı değerine hareket etmesi yerine, bulunduğu açı değerinden bağımsız olarak belirli bir açı, süre ve tur dönmesi istenebilir. Bu durumlarda aşağıdaki komutlar kullanılır.

run\_for\_degrees(miktar, hız) # Belirtilen açı miktarı kadar dönme

run\_for\_rotations(miktar, hız) # Belirtilen tur sayısı kadar dönme

run\_for\_seconds(miktar, hız) # Belirtilen süre miktarı kadar dönme

Örneğin motorun saat yönünde 90 derece 50 hızında hareket etmesi isteniyorsa aşağıdaki komut kullanılır.

motor.run\_for\_degrees(90,50)

Başka bir örnek vermek gerekirse, motorun 1.5 saniye saat yönünün tersine 50 hızında hareket etmesi isteniyorsa aşağıdaki komut kullanılır.

motor.run\_for\_seconds(1.5,-50)

Açı miktarı tam sayı değerleri alabilir. Tur sayısı ve süre ondalık değerler alabilir. Süre saniye cinsindendir. Pozitif değerler saat yönünde negatif değerler ise saat yönünün tersine hareketi sağlar. Süre pozitif tam sayı değerleri alabilir. Pozitif değerler hareketin yönünü değiştirmezken negatif değerler hareketi ters yöne çevirir.Rehber öğretmen yukarıda bulunan üç komutu da öğrencilerin deneyimlemesini sağar ve gerekli açıklamaları yapar.

|  |
| --- |
| Not |
| Daha önce sürüş tabanında kullanılan *set\_default\_speed* ve *set\_stop\_action* metotları kullanılarak motorlar için ön tanımlı hız ve durma şekli belirlenebilir. |

### Uygula: Belirli Bir Derece/Tur/Saniye Hareket Etme

Öğrenciler sürüş tabanında hareketli kolu inşa eder. Ardından rehber öğretmen, öğrencilerden sürüş tabanındaki hareket eden kolu büyük motor kullanılarak yukarı aşağı hareket ettirmeleri istenir. Bu hareket ritmik dans hareketleri şeklinde olmalıdır.

## TASARLA

Tasarla: Cisimleri Kenardan Ortaya Taşıyorum

Bu etkinlik için aşağıdaki şekilde gösterildiği gibi sürüş tabanı tarafından taşınmak üzere iki cisme ihtiyaç duyulacaktır. Öğrenciler robot tarafından taşınabilir nitelikte olmak kaydıyla farklı cisimler inşa edebilir.



*Resim 5.5 Taşınacak Cisimler*

Sürüş tabanı ve taşınacak cisimler aşağıdaki şekilde yerleştirilir. Çalışma alanı örnek olarak gösterilmiştir. Bu şekilde eliptik bir yüzey kullanılmak zorunda değildir. Şekilde görülen, sürüş tabanının başlangıç duruşudur. Bu etkinlikte robot kenarlardaki cisimleri orta kısımda toplayacaktır.



*Resim 5.6 Etkinlik Başlangıç*

Aşağıdaki resimde görüldüğü gibi robot görevini bitirdiğinde cisimler orta kısımda olacaktır. Robot ise cisimlerin arka kısmında şekildeki gibi durmalıdır.



*Resim 5.7 Etkinlik Bitiş*

Rehber öğretmen öğrencilerden öncelikle çözümleri için bir tasarım hazırlamaları gerektiğini söyler. Öğrenciler doğrudan çözümü inşa etmeye başlamamalıdır. Öncelikle çözüm üzerine düşünüp bir tasarısını oluşturmalıdır. Tasarım için aşağıda gösterilen iki adımlı bir süreç kullanılabilir.

Tanımlama: Bu adımda öğrenciler problemi ve bileşenlerini tanımlamalıdır. Kâğıt kalem alarak robotun yapması gerektiği hareketin krokisini çizebilirler. Ardından robot için bu hareketlerin ne anlama geldiğini ve hangi adımlardan geçerek bu görevi yerine getirebileceğini tanımlarlar.

Bu problemin çözümü için:

* Robot 90 derece sağa dönecek,
* Cisme kadar ilerleyecek ve cismi alıp geri geri gelerek orta kısma taşıyacak,
* Cismi bıraktıktan sonra 90 derece sağa dönecek ve yeterli miktarda geri gelecek,
* Robot yeniden 90 derece sola dönecek ve diğer cismi alıp bir miktar ileriye taşıyacak,
* Daha sonra 90 derece sağa dönerek cismi orta kısma taşıyıp bırakacak,
* Cismi bıraktıktan sonra robot bir miktar geri gelecek 90 derece sola dönecek,
* Cisim bir miktar daha geriye gidecek ve 90 derece sağa dönecek,
* Robot bir miktar ilerledikten sonra 90 derece sağa dönecek ve cisimlere yaklaşacak.

Dikkat edilmesi gereken noktalar:

1. Robotun sağa ve sola 180 derece dönmesi gerekecek bu dönüşler olabildiğince keskin bir şekilde ayarlanmalı,
2. Robotun cisimlere olan uzaklığı ölçülüp gerekli hareketler olabildiğince keskin şekilde yapılmalı,
3. Robot kolun aşağı ve yukarı hareketi cisimlerin boyutlarına göre ayarlanmalı ve bu hareket olabildiğince keskin şekilde yapılmalı.

Fikir üretme: Bu aşamada öğrencilerden tanımlamada belirlenen işlemlerin nasıl yapılacağına dair fikir üretmeleri beklenmektedir.

* 90 derece sağa ve sola dönüşler için robotun kendi merkezinde dönmesi sağlanmalıdır,
* 180 derecelik bir tekerlek dönüşü 90 derecelik robotun dönüşüne denk gelebilir. Bu değer denenerek keskin dönüş için uyarlanmalıdır.
* Robot kolun yukarı aşağı hareketi için gerekli açı değişim miktarları 90 dereceden küçük olmalıdır. Bu açılar denenerek bulunmalıdır. Bu iş için *move\_to\_position* ve *run\_for\_degrees* metotları kullanılabilir.
* Robotun cisme olan uzaklığı ölçülüp bulunmalıdır. Move komutu kullanılarak bu işlem yapılabilir. Bu işlemin keskin bir şekilde yapılması için denemeler yapılmalıdır.
* Diğer manevralar için de gerekli denemeler yapılmalıdır.

Tasarla aşamasında öğrenciler yukarıda anlatılan adımları birebir gerçekleştirmek zorunda değildir. Bu aşamadaki amaç problemin sınırlarının iyice çizilmesinin ardından, çözüm üzerine düşünmek onun bir tasarısını çıkarmaktır. İyi programcıların en fazla vakit ayırdığı etkinlik tasarıdır. Giriş seviyesindeki programcılar genel olarak çözüme doğrudan başlar ve deneme yanılma süreciyle ilerlemeye çalışır. Bu yeterince verimli bir yaklaşım değildir. Tüm bunlarla birlikte gerçek yaşamda program yazılırken tasarı ve üretim aşamaları zaman zaman iç içe geçebilir. Burada gösterildiği kadar ayrık bir şekilde ele alınmayabilir. Buna benzer şekilde tasarlama adımlarının her birinde öğrencinin mutlaka tanımlama ve fikir üretme adımlarını yapmasını beklemek gerçekçi olmayabilir. Fakat gene de problem çözümünden önce tasarımı hakkında düşünmesi öğrenciler için faydalı olacaktır. Rehber öğretmen tasarla adımını öğrencileri sıkmayacak ve aynı zamanda verimli olacak şekilde uygulamalıdır.

## ÜRET

Üret aşamasında öğrenciler aktif bir şekilde robotu ve programlama ortamını kullanarak çözüme ulaşmaya çalışırlar. Öğrenciler bu aşamada grup arkadaşı ile işbirlikli çalışmalıdır. Rehber öğretmen öğrencilerin sorularına doğrudan yanıtlar vermemelidir. Öğrencilerin çözümü kendi bulmaları ana hedeftir. Fakat onları çözüm konusunda düşündürecek şekilde yönlendirebilir. Bu aşamada rehber öğretmen kesinlikle tahta karşısına geçip çözümü öğrencilere göstermemelidir. Üret etkinliklerindeki amaç öğrencilerin üst seviye bilişsel etkinlikler göstermeleridir. Çözüme ulaşmak ikincildir. Bu yüzden çözüm yerine çözüm sürecindeki öğrenci etkinlikleri gözlenmelidir. Rehber öğretmen öğrencileri ileri seviye düşünme becerilerini tatbik edecek şekilde yönlendirmelidir.

## DEĞERLENDİR

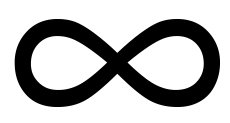
Günün sonunda öğrenciler ile halka oluşturulur ve aşağıdaki sorular tartışılır:

* Sürüş tabanında tekerlekler arasındaki mesafe artırılırsa move ve move\_tank komutları eski durumda ürettiği sonuçları aynen üretir mi?
* Verilen görevleri göz önünde bulundurduğunuzda en çok hangi görevde zorlandınız? Bu zorlukların üstesinden nasıl geldiniz? (Problemin çözümü için hangi stratejileri kullandınız ve neden bu stratejileri seçtiniz?) Yeteri kadar tartışma ortamı oluşmazsa, rehber öğretmen aşağıdaki soruları kullanarak tartışma ortamı yaratmaya çalışır.
  + Robot hareketlerini keskin bir şekilde yerine getirmekte zorlandınız mı?
  + Komutları akılda tutmakta veya doğru yazmakta zorlandınız mı?
  + Cisimleri Kenardan Orta Kısma Taşıyorum etkinliğinde hangi komutları kullandınız. Bunları kullanma sebepleriniz nelerdir?
* Kullandığınız yöntemler, bu sıkıntıları gidermekte başarılı oldu mu?
* Grup arkadaşınızla fikir ayrılığına düştüğünüz durumlar oldu mu ve bunların üstesinden gelmek için neler yaptınız?
* Grup arkadaşınızdan ne öğrendiniz?
* Bir sonraki derse daha verimli bir öğrenme için neleri yapıp neleri yapmamanız gerekir?

## İLAVE ETKİNLİK

Sonsuz Üzerinde Hareket

Bu etkinlikte robotun sonsuz benzeri bir şekilde hareket etmesi istenmektedir. Bunun için rehber öğretmen tarafından belirlenen iki nesne kullanılacaktır. Bu nesneler sonsuz işaretinin göz kısımlarını (beyaz renkli boşluklar) temsil etmektedir. Nesnelerin arasındaki mesafe 40-50 cm olabilir. Robotun bu nesnelerin etrafından sonsuz şekli çizerek ve dokunmadan hareket etmesi sağlanmalıdır. Bunun için örnek bir yol aşağıdaki resimde verilmiştir.



*Resim 5.8 Robot Yolu*

Programı oluşturmaya başlamadan önce grupların tasarlama adımı için yukarıda bir örneği verilen tanımlama ve fikir üretme süreçlerini gerçekleştirmeleri gerekir. Zaman kalması durumunda bu etkinlik gerçekleştirilebilir. İlave etkinlik bittiğinde fazladan zaman kalırsa iki nesne arasındaki mesafe artırılabilir veya azaltılabilir. Öğrenciler bu yeni durumlar için kodlarını değiştirmelidir.