# 10. Hafta – Renk Sensörü

## Haftanın Amacı:

Bu haftanın amacı, öğrencilerin renk (color) sensörünün çeşitli kullanım şekillerini kavrayarak sensörü farklı görevler için programlarken gerekli düzenlemeleri yapabilmelerini sağlamaktır. Ayrıca, öğrencilerin PID algoritması ile robotun farklı görevlerde daha verimli çalışmasını sağlamak için gerekli düzenlemeleri yapabilmelerini sağlamak da amaçlanmıştır.

## Haftanın Kazanımları:

* Öğrenciler renk sensörünün nasıl çalıştığını ifade edebilirler.
* Öğrenciler renk sensörünün ölçtüğü renk değerine göre robotun davranışını düzenleyebilirler.
* Öğrenciler renk sensörünün ölçtüğü ortam ışığı değerine göre robotun davranışını düzenleyebilirler.
* Öğrenciler renk sensörünün ölçtüğü yansıyan ışık değerine göre robotun davranışını düzenleyebilirler.
* Öğrenciler PID algoritma kavramını açıklayabilirler.
* Öğrenciler oluşturdukları programların daha verimli çalışması için robotun PID algoritması içeren program kodlarını oluşturabilirler.

## Kullanılacak Malzemeler:

Robot seti, bilgisayar ve çalışma alanı

## GÖZLE VE UYGULA

|  |
| --- |
| ***Rehber Öğretmen İçeriği –*** ***Renk Sensörü Hakkında*** |
| *Renk sensörünün üç temel görevi bulunmaktadır. Bunlardan ilki gördüğü cismin rengini belirlemektir. Siyah (black), mor (violet), mavi (blue), açık mavi (cyan), yeşil (green), sarı (yellow), kırmızı (red) ve beyaz (white) renkleri algılayabilir. Bunun yanında karşıda tanımlanan renklerden herhangi birinin olmadığını da algılayabilir.*  *Renk sensörünün ikinci görevi yansıyan ışığın yoğunluğunu ölçmektir. Sensörde üç adet ışık kayağı bulunmaktadır. Bu ışık kaynaklarından beyaz ışık gönderir ve karşıdan yansıyan ışığın yoğunluğunu ölçebilir. Bu değer 0 ile 100 arasında değişir. 0 hiç ışık yansımadığı anlamına gelir 100 ise en yüksek yoğunlukta ışık yansıdığını belirtir. Yansıyan ışık yoğunluğu yansıtıcı yüzeyin rengine ve mat veya parlak oluşuna göre değişir. Siyah renk ışığı büyük oranda soğurur ve yansıtmaz, beyaz renk ise ışığı büyük oranda yansıtır. Mat yüzeyler ışığı dağıtırken parlak yüzeyler ışığı yansıtır. Karşıdaki cismin rengini belirlerken veya karşıdaki cisimden yansıyan ışığın yoğunluğunu belirlerken sensörün cisme olan ideal uzaklığı 16 mm’dir. En iyi ölçümleri bu mesafede yapacaktır.*  *Renk sensörünün son görevi ortam ışığının yoğunluğunu algılamaktır. Ortam ışığının yoğunluğu 0 ile 100 arasında değerler alabilir. 0 ortamda ışık olmadığı, 100 ise en yüksek yoğunluklu ışık olduğu anlamına gelir. Dolayısı ile 0 en karanlık ve 100 en aydınlık değerdir.*  *Bu görevlere ek olarak sensörde bulunan ışıklardan da bahsetmek yerinde olacaktır. Sensörde üç adet ışık kaynağı bulunmaktadır. Bu ışık kaynaklarının parlaklığı ayrı ayrı programlanabilir.* |

### Gözle: Cismin Rengini Belirliyorum

Bu etkinlikte yerinde duran robotun renk sensörüne gösterilen cismin rengi belirlenecektir. Gösterilecek cisimlerin rengi sensörün tanıdığı renkler olan siyah, mor, mavi, açık mavi, yeşil, sarı, kırmızı ve beyaz renklerden biri olmalıdır. Bunun dışındaki cisimler şu aşamada tanımlanamayabilir.

Diğer sensörlerde olduğu gibi öncelikle renk sensörü tanımlanır. Bu durum aslında nesne yönelimli programlamaya göre Renk sensörü (ColorSensor) sınıfından bir nesne oluşturulması anlamına gelir. Renk sensörü sınıfı MicroPython programcıları tarafından önceden tanımlanmıştır. Programcılar bu sınıfı kullanarak bir nesne oluştururlar ve kullanırlar. Sınıf genel bir kalıp gibidir, nesnelerin nasıl olacağını belirler. Nesneler ise bu sınıfların sonucunda oluşan “gerçek” varlıklardır. Bu durumu önceden kullandığımız listeler üzerinden açıklayabiliriz. Python programcıları bir liste sınıfı oluşturmuştur. Bu sınıf içerisinde listelerin hangi değerleri alabileceği (veri) ve bu değerler ile neler yapılabileceği (metot) belirlenmiştir. Fakat sınıf bir tanımlar bütünüdür. Program yazıldığında genellikle pratik olarak kullanılmaz. Bu tanımların pratik olarak kullanılabilmesi için onlardan nesne veya nesneler oluşturulmalıdır. Programcılar liste sınıfından bir nesne oluşturarak listeleri kullanabilir. Programcılar tanımlanan sınıfları kullanmanın ötesine geçip kendi sınıflarını da tanımlayabilirler. Nesne yönelimli programlama bu dersin içeriğine dahil değildir. Buradaki bilgiler öğrencilerin nesne yönelimli programlamada kullanılan terminolojiye giriş yapması için anlatılmıştır. Sonuç olarak öğrencilerin “renk sensörü kullanılmadan önce tanımlanmalıdır” gibi bir ifade ile karşılaştıklarında aslında ColorSensor sınıfından bir nesne oluşturarak bu nesne ile gerekli işlemleri yaptıklarını bilmeleri yeterlidir. Bu iş için aşağıdaki komut kullanılabilir.

renk\_sensoru=ColorSensor(“B”)

Bu komut B portuna bağlı bir renk sensörü nesnesi oluşturulduğunu ve oluşturulan nesnenin isminin renk\_sensoru olduğunu söylemektedir. Nesnenin isminin bu şekilde olması zorunlu değildir istenen isim verilebilir. Renk sensörünün karşısındaki cismin rengini belirlemek için get\_color metodu kullanılabilir. Bu metot algılanan rengin İngilizcesini metin olarak döndürür. Bir renk algılanmadığı durumda None değerini döndürür. Bu görev için gerekli kod aşağıda verilmiştir. Rehber öğretmen öğrencilere kodu açıklayarak anlatır.

hub = PrimeHub()

renk\_sensoru=ColorSensor("B")

while True:

    renk=renk\_sensoru.get\_color()

    if renk=="black":

        hub.light\_matrix.write("Siyah")

    elif renk=="violet":

        hub.light\_matrix.write("Mor")

    elif renk=="blue":

        hub.light\_matrix.write("Mavi")

    elif renk=="cyan":

        hub.light\_matrix.write("Acik Mavi")

    elif renk=="green":

        hub.light\_matrix.write("Yesil")

    elif renk=="yellow":

        hub.light\_matrix.write("Sari")

    elif renk=="red":

        hub.light\_matrix.write("Kırmızı")

    elif renk=="white":

        hub.light\_matrix.write("Beyaz")

    else: # None değerini kontrol eder

        hub.light\_matrix.write("Renk Yok")

    wait\_for\_seconds(1)

### Gözle: İleri ve Geri Komutunu Renkler ile Veriyorum

Bu etkinlikte robota renkler aracılığı ile ileri ve geri 10 cm gitme komutları verilecektir. Robota sarı renk gösterildiğinde 10 cm ileri gidip duracaktır. Robota kırmızı renk gösterildiğinde 10 cm geri gidip duracaktır. Yeni ileri veya geri komutu robot durduktan sonra verilmelidir. Robot her yeni komutta (sarı veya kırmızı) gerekli görevi yerine getirmelidir.

Get\_color metodu renk sensörünün okuduğu renk değerini anlık olarak döndürür. Bazı durumlarda anlık renk değeri okumak yerine yeni bir renk algılanana kadar bekleme işlemi yapılması gerekebilir. Bunun için wait\_for\_new\_color olayı kullanılır. Bu olay sayesinde renk sensörü yeni bir renk algılayana kadar bekler ve aynı zamanda yeni algılanan rengin değerini döndürür. Bu etkinlik için wait\_for\_new\_color olayı kullanılabilir. Aşağıda etkinliğin çözümüne yönelik örnek bir kod verilmiştir. Rehber öğretmen kodu öğrencilere açıklayarak anlatır.

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

renk\_sensoru=ColorSensor("B")

while True:

    renk=renk\_sensoru.wait\_for\_new\_color()

    if renk=="yellow":

        surus\_tabani.move(10,"cm",0,50)

    elif renk=="red":

        surus\_tabani.move(10,"cm",0,-50)

### Uygula: Renk Sensörü ile Programlanabilen Robot

Bu etkinlikte de öğrenciler renkler ile robotun nasıl hareket edeceğini programlayacaktır. Sarı renk robotun 10 cm ileri gitmesini sağlar, kırmızı renk robotun 10 cm geri gitmesini sağlar, yeşil renk robotun kendi ekseninde sola dönmesini sağlar ve mor renk robotun kendi ekseni etrafında sağa dönmesini sağlar. Fakat bu sefer robot her bir hareket sonunda kodlanmayacaktır. Öncelikle robota hangi hareketi yapması gerektiğini söyleyen komutlar verilecektir. Siyah renk gösterildiğinde komut verme işleminin bittiği anlaşılacaktır ve robot verilen komutları yerine getirecektir. Siyah renk ile verilen komutlar bittikten sonra robot tekrar tekrar yeni hareket komutlarını kabul ederek siyah renk ile yeniden harekete başlamalıdır. Rehber öğretmen etkinliği açıkladıktan sonra öğrencilerin gerekli çözümü yapmalarını sağlar. Rehber öğretmen için örnek bir kod aşağıda verilmiştir.

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

renk\_sensoru=ColorSensor("B")

komutlar=[]

renk= None

while True:

    if renk!="black":

        renk=renk\_sensoru.wait\_for\_new\_color()

    else:

        for renk in komutlar:

            if renk=="yellow":

                surus\_tabani.move(10,"cm",0,50)

            elif renk=="red":

                surus\_tabani.move(10,"cm",0,-50)

            elif renk=="green":

                surus\_tabani.move(180,"degrees",-100,50)

            elif renk=="violet":

                surus\_tabani.move(180,"degrees",100,50)

        komutlar.clear()

        renk=None

    if renk!=None:

        komutlar.append(renk)

|  |
| --- |
| Not |
| Bu kod ile aynı hareket komutu iki kere verilmek isteniyorsa istenen renk gösterilip çekilmeli ardından bir daha gösterilmelidir. |

### Gözle: Siyah Rengi Görene Kadar İlerleyen Robot

Bu etkinlikte robot renk sensöründe siyah rengi görene kadar ilerlemelidir. Siyah rengi görene kadar koşulunu belirlemek için get\_color ve wait\_for\_new\_color kullanılabilir. Fakat MicroPython doğrudan bir renk algılanana kadar bekle isimli bir olaya sahiptir. Bu komutun kullanımı aşağıda gösterilmiştir.

wait\_until\_color(renk)

Bu komuta parametre olarak girilen renk değeri algılanana kadar renk sensörü bekleyecektir. Buradaki renk parametresi çift tırnak içerisinde girilmelidir. Görev için aşağıdaki kod kullanılabilir.

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

renk\_sensoru=ColorSensor("B")

surus\_tabani.start(0,30)

renk\_sensoru.wait\_until\_color("black")

surus\_tabani.stop()

### Uygula: Renk Kodları ile Harekete Başlayan Robot

Bu etkinlikte de robot siyah rengi görene kadar ilerleyecektir. Fakat robot program çalışır çalışmaz ilerlemeye başlamamalıdır. Robotun hareketine başlayabilmesi için öncelikle renk kilidinin açılması gerekir. Renk kilidi mavi, yeşil ve sarıdır. Bu renkler sırasıyla renk sensörü tarafından algılanmadığı müddetçe hareket başlamaz. Rehber öğretmen öğrencilerin gerekli kodu yazmalarını sağlar. Aşağıda rehber öğretmen için hazırlanmış örnek bir kod bulunmaktadır.

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

renk\_sensoru=ColorSensor("B")

renk\_sensoru.wait\_until\_color("blue")

renk\_sensoru.wait\_until\_color("green")

renk\_sensoru.wait\_until\_color("yellow")

surus\_tabani.start(0,30)

renk\_sensoru.wait\_until\_color("black")

surus\_tabani.stop()

### Gözle: Ham Renkleri Öğreniyorum

Doğadaki bütün renkler kırmızı (**R**ed), yeşil (**G**reen) ve mavi (**B**lue) rengin karışımı şeklinde belirlenebilir. Örneğin kırmızı ve yeşil renk karıştırıldığında sarı renk elde edilir. Turuncu ise kırmızı ve sarı rengin karışımıdır. Buna RGB renk modeli denilir. MicroPython ile bir nesnenin renginde hangi yoğunlukta kırmızı, yeşil ve mavi renklerin bulunduğu öğrenilebilir. Bunun için get\_red (kırmızı için), get\_green (yeşil için) ve get\_blue (mavi için) metotları kullanılır. Bu metotların her biri 0-1024 arasında değer üretir. 0 değeri ilgili renk yoğunluğunun bulunmadığını gösterirken 1024 değeri o rengin yoğunluğunun en üst seviyede olduğunu gösterir. Örneğin tam olarak kırmızı renkteki bir nesne için get\_red metodu 1024 değeri döndürürken, get\_green ve get\_blue metotları 0 değeri döndürecektir. Uluslararası RGB sisteminde ve Spike Prime sözcük blokları kullanıldığında bu değerler 0-255 aralığındadır. Fakat MicroPython için bu değerler 0-1024 olacak şekilde standart dışı tanımlanmıştır.

Rehber öğretmen aşağıdaki kodu öğrenciler ile paylaşır. Öğrencilerin farklı renkte cisimler kullanarak programın çıktılarını gözlemelerini sağlar. Bu iş için Spike Prime seti ile gelen Lego tuğlaları kullanılabilir.

renk\_sensoru=ColorSensor("B")

while True:

    kirmizi=renk\_sensoru.get\_red()

    yesil=renk\_sensoru.get\_green()

    mavi=renk\_sensoru.get\_blue()

    print(kirmizi, yesil, mavi)

    wait\_for\_seconds(0.5)

Kod çalıştırıldığında tuğlaların renklerinin mutlak kırmızı (get\_red için 1024 ve diğerleri için 0), mutlak yeşil (get\_green için 1024 ve diğerleri için 0) ve mutlak mavi (get\_blue için 1024 ve diğerleri için 0) olmadıkları anlaşılacaktır. Kırmızı renkli tuğlada, ana renk olan kırmızının değeri diğerlerine göre daha yoğun olmakla birlikte, yeşil ve mavinin değerleri 0 değildir. Programın çalıştırıldığı koşullarda kırmızı için 650 (get\_red()), yeşil için 190 (get\_green()), mavi için 270 (get\_blue()) yaklaşık değerleri elde edilmiştir. Benzer durum diğer renkler için de söz konusudur.

### Uygula: Turuncu Rengi Algıladığında Bip Sesi Çalan Robot

Bu etkinlikte öğrencilerden turuncu rengi algıladığında bip sesi çalan bir robot programı yazmaları beklenmektedir. Renk sensörü turuncu rengi doğrudan algılayamamaktadır. Öğrenciler get\_red, get\_green ve get\_blue metotlarını kullanarak turuncu rengi tanımlayabilirler. Bunun için turuncu renkli bir cisim alınmalıdır ve bu cismin kırmızı, yeşil ve mavi yoğunluk değerleri bulunmalıdır. Ardından bu değerlere yakın bir aralıkta bulunan değerler turuncu olarak tanımlanmalıdır. Programın çalıştırıldığı koşullarda kullanılan turuncu nesne içi kırmızı=700, yeşil=320 ve mavi=300 yaklaşık yoğunluk değerleri bulunmuştur. Bu sebeple turuncu için 690<kırmızı<710, 310<yeşil<330, ve 290<mavi<310 aralıkları kullanılmıştır. Bu değerler uygulama koşullarında öğrenciler için farklılık gösterecektir. Öğrenciler kendi değerlerini bulmalıdır. Bulunan aralıklar için yazılmış örnek kod aşağıda verilmiştir.

renk\_sensoru=ColorSensor("B")

while True:

    kirmizi=renk\_sensoru.get\_red()

    yesil=renk\_sensoru.get\_green()

    mavi=renk\_sensoru.get\_blue()

    if 690<kirmizi<710 and 310<yesil<330 and 290<mavi<310:

        hub.speaker.beep()

|  |
| --- |
| Not |
| RGB renk modeli renk algılanması işlemi için verimli değildir. Bu iş için HSV adında başka bir renk modeli daha verimli sonuçlar üretir. Fakat ders kapsamında bu modelin bilinmesi gerekli değildir. Öğrencilerin RGB modeli ile renk algılama kodlarının verimsiz olduğunu bilmesi yeterlidir. |

### Gözle: Renk Sensöründeki Işıkları Yakıp Söndürüyorum

Bu etkinlikte renk sensöründeki üç ışığın 0.5 saniye aralıklarla yakılıp söndürülmesi sağlanacaktır. Daha önce söylendiği gibi renk sensöründe üç adet ışık kaynağı bulunur. Bunları aynı anda kodlamak için light\_up\_all metodu kullanılır. Aşağıda light\_up\_all metodunun kullanımı gösterilmiştir.

light\_up\_all(parlaklık)

Parlaklık değeri 0 (en karanlık) ve 100 (en parlak) arasında değişir. Işık kaynaklarının parlaklık değerleri ayrı ayrı kodlanmak isteniyorsa aşağıdaki komut kullanılır. Bu komut ile her bir ışığın parlaklığı 0-100 arasında olacak şekilde ayrı ayrı belirlenebilir.

light\_up(ışık 1 parlaklık, ışık 2 parlaklık, ışık 3 parlaklık)

Aşağıda ışıkları yakıp söndürme görevi için kullanılabilecek örnek bir kod verilmiştir. Rehber öğretmen bu kodu öğrencilere açıklayarak anlatır.

renk\_sensoru=ColorSensor("B")

while True:

    renk\_sensoru.light\_up\_all(100)

    wait\_for\_seconds(0.3)

    renk\_sensoru.light\_up\_all(0)

    wait\_for\_seconds(0.3)

### Uygula: Işık Kaynakları Sırayla Hareket Ediyor

Bu etkinlikte ışık kaynakları sırayla dönme hissi verecek şekilde yakılmalıdır. Bu görev için aşağıda örnek bir kod verilmiştir. Öğrenciler farklı şekillerde de dönme hareketini sağlayabilir. Bu kodun aynısını yazmak zorunda değildir.

renk\_sensoru=ColorSensor("B")

while True:

    renk\_sensoru.light\_up(100,0,0)

    wait\_for\_seconds(0.1)

    renk\_sensoru.light\_up(0,100,0)

    wait\_for\_seconds(0.1)

    renk\_sensoru.light\_up(0,0,100)

    wait\_for\_seconds(0.1)

### Gözle: Ortam Işığının Yoğunluğunu ölçüyorum

Ortam ışığının yoğunluğunu ölçmek için get\_ambient\_light metodu kullanılır. Bu metot ile ölçüm yapılırken renk sensöründe bulunan ışık kaynaklarından ışık gönderilmez. Ortam ışığı 0 ile 100 arasında değerler alabilir. 0 ortam ışığı yoğunluğunun olmadığını (en karanlık) 100 ise ortam ışığı yoğunluğunun en üst seviyede (en aydınlık) olduğunu gösterir. Rehber öğretmen aşağıdaki kodu öğrenciler ile paylaşır ve farklı aydınlık değerleri için programın çıktılarını gözlemlemelerini sağlar. Bunun için cep telefonlarının el feneri özelliği kullanılabilir.

renk\_sensoru=ColorSensor("B")

while True:

    print(renk\_sensoru.get\_ambient\_light())

    wait\_for\_seconds(0.2)

### Uygula: Hava Aydınlandığında Çalan Alarm

Bu etkinlikte her hava aydınlandığında robotun alarm çalması istenmektedir. Robot hava aydınlandığında dört defa alarm çalmalıdır (bip sesi) ve renk sensörünün ışıklarını yakıp söndürmelidir. Dört defa bu işlem tekrarlandıktan sonra alarm susmalıdır. Hava kararmasının ardından yeniden hava aydınlandığında alarm yeniden söylendiği şekilde devreye girmelidir. Böylece her hava aydınlandığında alarm devreye girecektir. Bu görev için örnek bir kod aşağıda verilmiştir.

renk\_sensoru=ColorSensor("B")

alarm\_kurulu=True

while True:

    if renk\_sensoru.get\_ambient\_light()>50 and alarm\_kurulu==True:

        for i in range (4):

            for i in range(60,110,20):

                renk\_sensoru.light\_up\_all(100)

                hub.speaker.beep(i)

                renk\_sensoru.light\_up\_all(0)

                wait\_for\_seconds(0.2)

        alarm\_kurulu=False

    else:

        if alarm\_kurulu==False:

            alarm\_kurulu=True

### Gözle ve Uygula: Yansıyan Işık Yoğunluğunu Belirleyen Faktörler

Yansıyan ışık ölçülürken renk sensörü üzerindeki üç ışık kaynağından beyaz ışık gönderir. Karşıdaki cisme (eğer varsa) çarpan ışık yansıyarak renk sensörüne geri döner. Renk sensörü yansıyan ışığın yoğunluğunu ölçebilir. Yansıyan ışığın yoğunluğunu ölçmek için get\_reflected\_light metodu kullanılır. Bu metot 0 ile 100 arasında değerler üretir. 0 yansıyan ışığın yoğunluğunun olmadığı ve 100 ise yansıyan ışığın yoğunluğunun en yüksek seviyede olduğunu söyler.

Yansıyan ışığın yoğunluğunu belirleyen çeşitli parametreler vardır. Bunlardan ilki mesafedir. Yansıyan ışık yoğunluğu mesafe ile ters orantılıdır. Renk sensörü karşısındaki cisme ne kadar uzaksa yansıyan ışık yoğunluğu o kadar azalır. İkinci parametre yüzeyin cinsidir. Mat ve pürüzlü yüzeyler ışığı daha az yansıtırken pürüzsüz ve parlak yüzeyler ışığı daha iyi yansıtır. Son parametre ise cismin rengidir. Siyah gibi rengi daha çok soğuran (emen) renkler ışığı daha az yansıtırken beyaz gibi ışığı fazla soğurmayan renkler ışığı daha iyi yansıtır. Işık daha az yansıtıldığında ölçülen ışık yoğunluğu daha az olurken iyi yansıtılan yüzeylerde ışık yoğunluğu daha fazla olacaktır.

Bu etkinlikteki amaç öğrencilerin farklı yüzeyler ve mesafeler için yansıyan ışık yoğunluğunu deneyimlemeleridir. Rehber öğretmen aşağıdaki kodu öğrencilerle paylaşır ve farklı mesafeler ve yüzeyler için kodun çıktısını gözlemelerini sağlar. Bunun ardından öğrencilerden yansıyan ışığın şiddetini belirleyen faktörleri sınıfça tartışmalarını sağlar. Tartışma sonucunda açık noktalar bulunursa rehber öğretmen gerekli bilgilendirmeyi yapar.

renk\_sensoru=ColorSensor("B")

while True:

    print(renk\_sensoru.get\_reflected\_light())

    wait\_for\_seconds(0.5)

Bu görev için (i) farklı mesafelerden, (ii) aynı renkteki Lego parçalarından delikli ve düz olanlarından, (iii) siyah ve beyaz yüzeylerden, (iv) farklı renkteki yüzeylerden ve (v) mat ve parlak yüzeylerden yansıyan ışık yoğunluğu ölçülebilir.

### Gözle ve Uygula: PID Algoritması ile Çizgi Takip Eden Robot

Bu etkinlikte PID algoritması ile çizgi takip eden robot programı yazılacaktır. Bu ders kapsamında dördüncü haftada çizgi takip programı yapılmıştır. Bu program fazla zig zag yapacak şekilde hareket eder ve eğimli çizgileri takip etmekte zorlanır. Yani program yeterince verimli çalışmaz. Bu etkinlikte robot aşağıdaki şekilde gösterilen eğri bir çiziyi takip edecektir.



*Resim 10.1 Çizgi Takip Eden Robot Etkinliği*

Günlük yaşamda birçok alanda gerçekte var olan bir değerin belirlenen hedef değere ulaşıp burada olabildiğince kalmasını sağlamak önemlidir. Örneğin kışın soğuk havalarda termostat sıcaklığı 23 dereceye ayarlanan bir kombi oda sıcaklığını 23 derece civarında tutmaya çabalar. Oda sıcaklığı 23 derecenin altında ise kombi çalışır. Oda sıcaklığı 23 derece olunca termostat devreye girer ve kombiyi kapatır. Fakat peteklerde sıcak su olduğu için oda ısınmaya devam eder oda sıcaklığı 23 dereceyi geçer. Kaloriferlerdeki sıcak su yeterince ısı sağlayamadığında oda yavaş yavaş soğur. 23 derecenin altında termostat kombiyi yeniden çalıştırır. Fakat kombi peteklerdeki suyu ısıtana kadar oda sıcaklığı 23 derecenin altına düşmüş olur. Bu şekilde oda sıcaklığı 23 derecenin belirli bir miktar altı ve üstü aralığında gelip gider. Ev kullanımında oda sıcaklığının bu aralıkta salınım yapması önemli bir probleme sebep olmaz fakat hassas işler gerektiren durumlarda bu salınımın mümkün mertebe az olması gerekir. Bu tip durumlarda PID isimli bir algoritma kullanılabilir.

Çizgi izleyen robotta ışık sensörünün ışığının bir kısmının siyah çizgide bir kısmının ise beyaz çizgide olması istenir. Aşağıdaki şekilde gösterildiği gibi bu durumda yansıyan ışık yoğunluğu yaklaşık olarak 60 olmaktadır. Bu değer ortamdaki ışık miktarına, yüzey için kullanılan kâğıda ve kâğıt üzerindeki renklerin niteliğine göre değişiklik gösterebilir.

A picture containing indoor, device

Description automatically generated

*Resim10.2 Sensörün Bulunması İstenen Bölge*

Saat yönünde bir miktar dönme hareketi ile yansıyan ışık yoğunluğu 60’ın üstüne çıkar ve tersinde bir hareket ile 60’ın altına iner. Çizgiyi takip edebilmesi için robotun olabildiğince yansıyan ışık miktarını 60 civarında tutmaya çalışması gerekir. Daha önce söylendiği üzere bu tip bir görev için PID kullanılabilir.

PID algoritmasının **P**roportional (oransal), **I**ntegral (integral) ve **D**erivative (türev) olmak üzere üç bileşeni bulunur. Algoritmanın ismi bileşen isimlerinin kısaltmasından oluşur. Aslında integral ve türev ileri matematiğin konusudur. Fakat burada kolaylık olması açısından bu kavramlar basitçe ele alınacaktır. PID algoritmasının üç bileşeni de hataya bağlıdır. Hata gözlenen (gerçekleşen) ve istenen değer arasındaki farktır.

Hata= Gerçekleşen – İstenen

Algoritmanın bileşenleri hataya farklı şekillerde tepki vererek hatanın istenen seviyede kalıp mümkün mertebe sıfıra yakın olmasına çalışır. Algoritmanın oransal kısmı hatanın şu anki haliyle ilgilenir ve ona oransal bir tepki verir. Aşağıda gösterildiği gibi Hata Kp isimli katsayı ile çarpılarak tepki oluşturulur. Bu katsayılar farklı koşullarda farklı tepkiler oluşturmak için kullanılır. Çizgiyi izlerken kullanılacak tepki ile oda sıcaklığını belirli bir sıcaklıkta tutmak için kullanılan katsayılar farklılaşacaktır.

Kp x Hata

Örneğin ortamdan ansıyan ışık değeri 80 olsun, bu durumda 80-60=20’lik bir hata bulunur. Yani robot siyah şeritten uzaklaşmıştır. Bu hata oranında robotun yönü saat yönünün tersine çevrilir ve hata telafi edilmeye çalışılır.

Gerçek yaşamda bazı durumlarda oransal tepki hatayı telafi etmekte yeterli olsa da birçok durumda yeterli olamaz. Robot zig zaglar çizerek ilerler. Bu da onun 60’ın altında ve üstünde değerler aldığını gösterir. Her bir zig zag hareketinde yeni hatalar oluşmakta ve bu hataların toplamı robotu hedeften saptırmaktadır. Burada integral devreye girer. Hataların toplamı oranında düzeltme yapılması gerekir. İntegral ile geçmişte yapılan hatalar telafi edilmeye çalışılır.

I=Hata Toplamı

Programlama açısından bakılacak olursa aşağıdaki formül bulunacaktır. Bu formül bir döngü içerisinde düşünülmelidir. Her döngü adımında yeni hata integrale eklendiği için toplam hata elde edilir. Bunun için integralin ilk değerinin 0 olarak belirlenmiş olması gerekir. Aksi taktirde toplam hata tam olarak bulunamaz.

I = I + Hata

Oransal ve integralin toplamı ile şu anki hata ve geçmişte yapılan hatalar giderilmeye *çalışılmıştır*. Fakat bazen bu işi yaparken uç durumlar oluşabilir. Bu da verilen tepkinin gereğinden keskin olmasına neden olabilir. Bu durumda gelecekte yapılma ihtimali olan hataların göz önünde bulundurulması gerekir. Gelecekte aşırı bir tepki verilme ihtimali varsa bu dengelenmelidir. Dengeleme görevi türeve düşer. Gelecekteki hata değerini tam olarak belirlemek mümkün değildir. Fakat gelecekteki hata şu anki hata ve bir önceki hata göz önünde bulundurarak tahmin edilebilir. Şu anki hatadan bir önceki hata çıkarılır. Bu değer eğer büyükse son iki hata arasındaki fark büyük olduğundan bir sonraki hatanın da büyük olma ihtimali vardır. Yani aşırı tepki ihtimali doğmuştur. Aşağıda formülü verilen türev ile bu aşırılık telafi edilmeye çalışılır.

T=Güncel Hata – Son Hata

Oluşturulacak tepki bu üçlü bileşen tarafından belirlenir. Fakat üç bileşen tepkiyi oluştururken doğrudan alınmaz. Bunlar birer kat sayı ile çarpılarak tepki oluşturulur. Bu katsayılara sırasıyla Kp (P için), Ki (I için) ve Kd (D için) ismi verilmiştir.

Tepki=Kp x P + Ki x I + Kd x D

Bu görev için gerekli kod aşağıda verilmiştir. Kodda bulunan kat sayılar ve istenen değer aynı ortamda farklı aydınlanma miktarlarına göre bile farklılık gösterir. Hava aydınlıkken çalışan kod hava karardığında tam olarak çalışmayabilir. Katsayılar ve istenen değer öğrenciler ve rehber öğretmen tarafından ayarlanmalıdır.

hub = PrimeHub()

renk\_sensoru=ColorSensor("B")

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

Kp=1.75   # Katsayılar farklı ortamlarda

Ki=0.015  # farklı değerler alabilir.

Kd=0.3    # Kendi ortamınıza göre düzenleyiniz.

P=0

I=0

D=0

son\_hata=0

istenen=60  # Bu değer ortama bağlı olarak değişebilir.

while True: # Kendi ortamınıza göre düzenleyiniz.

    gerceklesen=renk\_sensoru.get\_reflected\_light()

    hata=istenen-gerceklesen # Hata robot hareket yönüne

    P=hata                   # negatif olarak yansıyacaktır.

    I=I+hata                 #Bu yüzden istenen-gerceklesen alinmistir

    D=hata-son\_hata

    Tepki=int(Kp\*P+Ki\*I+Kd\*D) # Tepki tam sayıya çevrildi

    surus\_tabani.start(Tepki,30) # start metodu tam sayı istediği için

    son\_hata=hata

Rehber öğretmen öncelikle bu etkinlikte ne yapılacağını öğrencilere anlatır. Ardından öğrencilere PID konusunu anlatır. Anlattığı konuya göre gerekli programı yazar ve programı yazarken açıklar. Tüm bu süreç bittikten sonra öğrencilerden istenen kodu kendilerinin yazmasını sağlar.

|  |
| --- |
| Not |
| Çalışma alanındaki eğri çizgi için Hafta10\_PID\_Egri\_Mat ismindeki dosyada bulunan sayfalar A4 kâğıdına çıktı alınır. Çıktı alındıktan sonra sayfalar sırası ile birleştirilerek çalışma alanı oluşturulur. Çalışma alanı oluşturulurken iki A4 kâğıdı arasında boşluklar kalabilir. Bu boşluklar kesilerek siyah çizgilerin aralıksız bir şekilde birbirini takip etmesi sağlanır. |

### Uygula: Dikdörtgen Şeklinde Çizgiyi Takip Eden Robot

Bu etkinlikte PID algoritması kullanarak dikdörtgen şeklinde bir çizgiyi takip eden robot programı yazılacaktır. Robot dörtgenin köşelerinde keskin dönüşler yapmalıdır. Yani geçmişteki ve gelecekteki hatalara nazaran güncel hata ön plandadır. Bu yüzden Kp katsayısı yüksek tutulmalıdır. Kp gereğinden fazla olursa robot çizgi takibini bırakıp çizgiyi geçebilir, gereğinden az olursa robot çizgiyi bulamayabilir veya geç bularak kendi etrafında dönmeye başlayabilir. Bu yüzden Kp katsayısı çok iyi ayarlanmalıdır. Kp’nin ayarlanmasının ardından deneme yanılma yöntemi ile diğer katsayılar da ayarlanabilir. Rehber öğretmen öğrencilere katsayılar hakkındaki bu bilgileri vermeden onların kod yazmalarını sağlar. Ardından sınıfça tartışma yaparak katsayıların nasıl ayarlanabileceğini öğrencilerin anlamalarını sağlar.

hub = PrimeHub()

renk\_sensoru=ColorSensor("B")

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

Kp=2.55   # Katsayılar farklı ortamlarda

Ki=0.001  # farklı değerler alabilir.

Kd=0.016  # Kendi ortamınıza göre düzenleyiniz.

P=0

I=0

D=0

son\_hata=0

istenen=60  # Bu değer ortama bağlı olarak değişebilir.

while True: # Kendi ortamınıza göre düzenleyiniz.

    gerceklesen=renk\_sensoru.get\_reflected\_light()

    hata=istenen-gerceklesen # Hata robot hareket yönüne

    P=hata                   # negatif olarak yansıyacaktır.

    I=I+hata                 #Bu yüzden istenen-gerceklesen alinmistir

    D=hata-son\_hata

    Tepki=int(Kp\*P+Ki\*I+Kd\*D) # Tepki tam sayıya çevrildi

    surus\_tabani.start(Tepki,30) # start metodu tam sayı istediği için

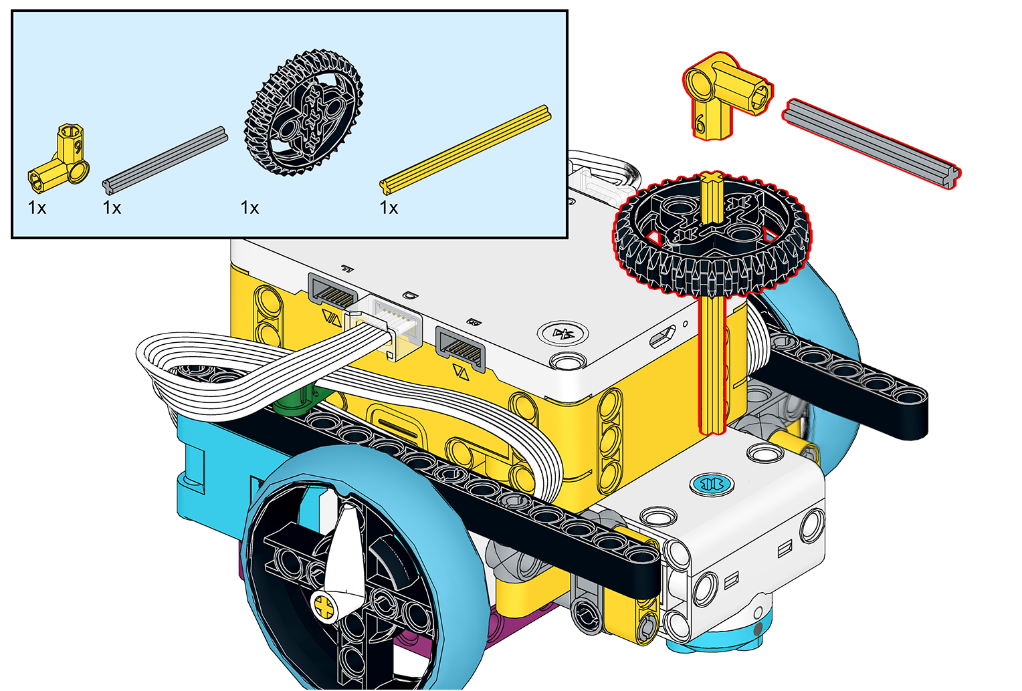
    son\_hata=hata

|  |
| --- |
| Not |
| Çalışma alanı için ekte Hafta10\_PID\_Dortgen\_Mat dosyasında bulunan A4 boyutundaki belgenin çıktısı alınır. Bu sayfalar bir bant aracılığı ile birleştirilir. Birleştirme esnasında çizgiler arasında boşluklar bulunabilir. Boşluk bulunan kısımlar makasla kesilerek kesintisiz bir dikdörtgen oluşturulması sağlanır. Ayrıca, program dörtgen çalışma alanı üzerinde denenirken robot köşelerde kağıtların dışına çıkacak şekilde taşıyorsa, köşelere beyaz A4 kâğıdı konulmalıdır. Aksi halde çalışma masasının rengi programın çalışmasını etkileyebilir. A3 çıktısı almak için uygun yazıcısı olanlar Hafta10\_PID\_Dortgen\_Mat \_A3 dosyasının çıktısını alıp doğrudan kullanabilirler. |

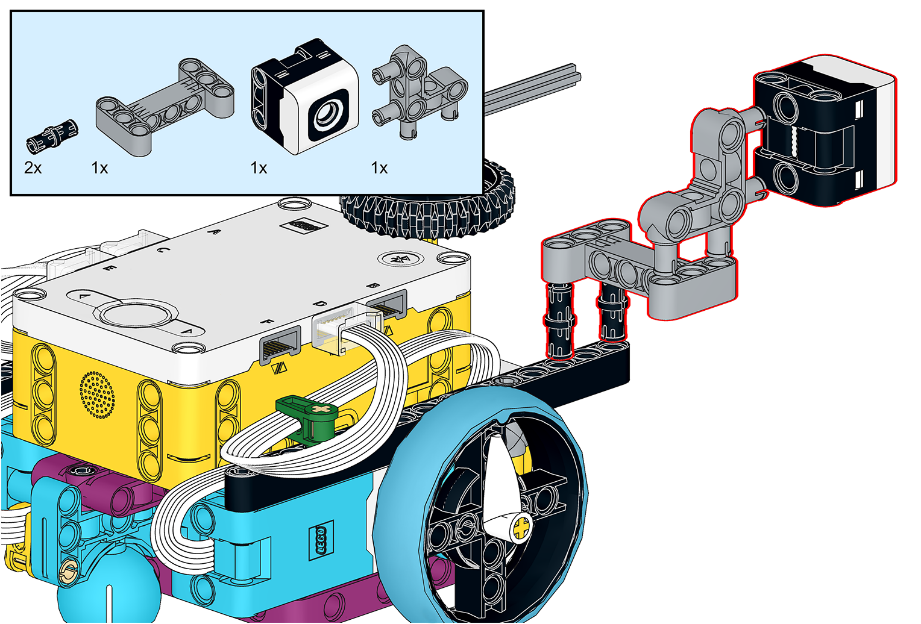
## TASARLA

### Lego Parçasıyla Robotu Kodluyorum

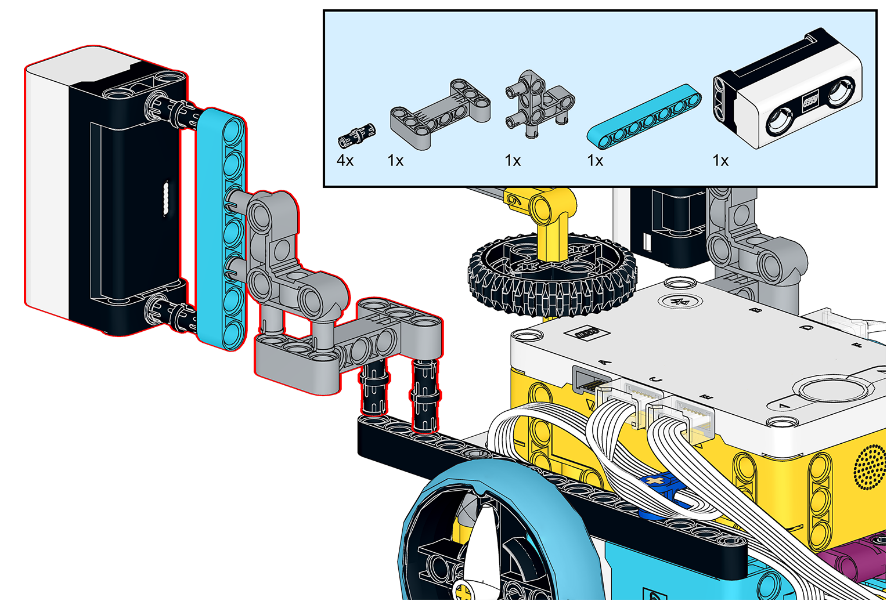
Bu etkinlikte renk sensörü, mesafe sensörü, kuvvet sensörü ve hub özellikleri birlikte kullanılacaktır. Etkinlik için kullanılacak robotun inşası aşağıda gösterilmiştir.



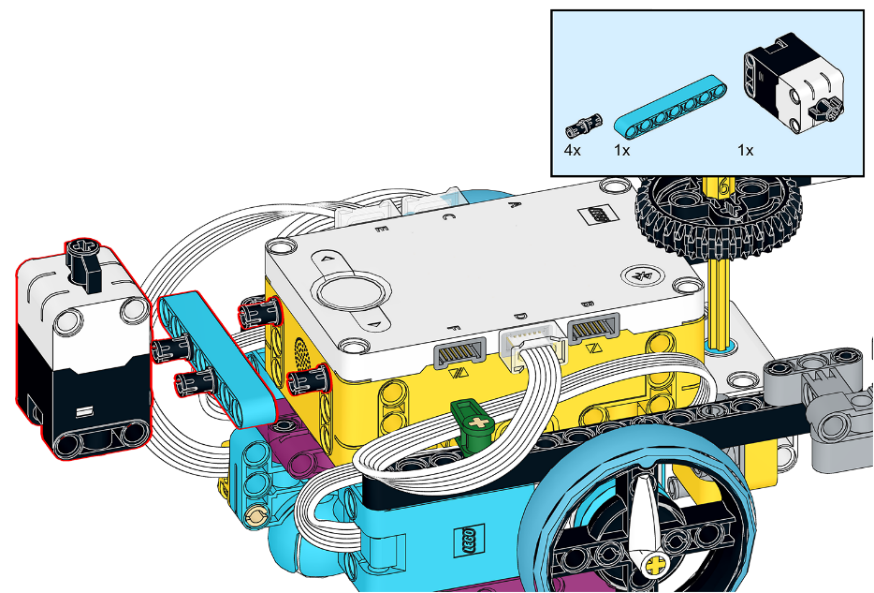
*Resim 10.3 Etkinlik İçin Robotun Düzenlenmesi Birinci Adım*



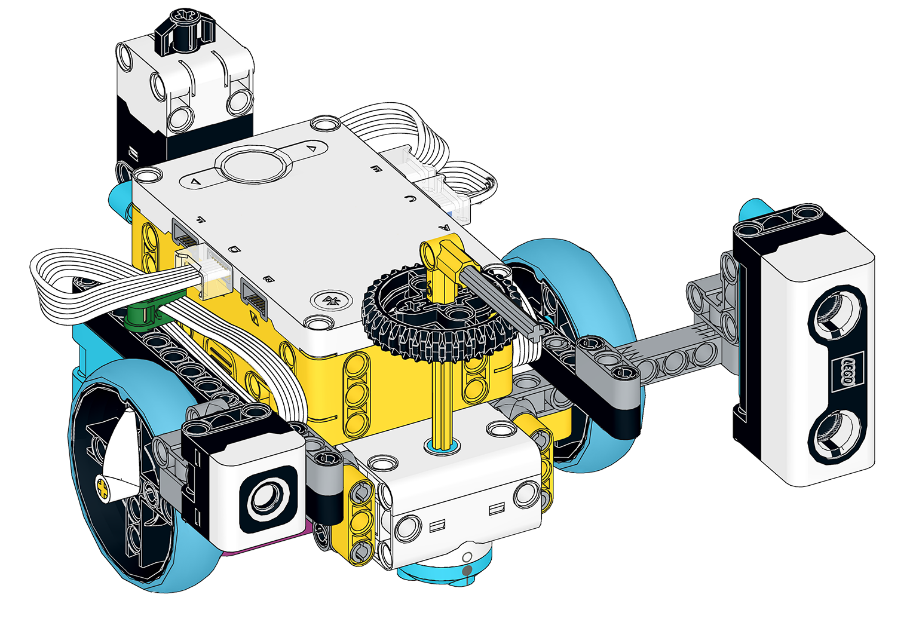
*Resim 10.4 Etkinlik İçin Robotun Düzenlenmesi İkinci Adım*



*Resim 10.5 Etkinlik İçin Robotun Düzenlenmesi Üçüncü Adım*



*Resim 10.6 Etkinlik İçin Robotun Düzenlenmesi Dördüncü Adım*



*Resim 10.7 Etkinlik İçin Kullanılacak Robot*

Robotun hareketi aşağıda örneği verilecek bir Lego parçası ile şu kurallara uygun olacak şekilde kodlanacaktır:

1. Lego parçası önce renk sensörünün ve ardından mesafe sensörünün önünden çok yakın olacak şekilde geçerse bu sağa dön komutu olacaktır.
2. Lego Parçası önce mesafe sensörünün ve ardından renk sensörünün önünden çok yakın olacak şekilde geçerse bu sola dön komutu olacaktır.
3. Lego parçası mesafe sensörünün yakınından başlayıp, ondan uzaklaşırsa bu ileri git komutu olacaktır.
4. Lego parçası mesafe sensörünün uzağından başlayıp yakınına giderse bu geri git komutu olacaktır.
5. İleri-geri ne kadar gidileceğini ve sağa-sola ne kadar dönüleceğini büyük motora bağlı olan değer kolu belirleyecektir. Değer kolu saat yönünün tersine döndükçe artan değerler elde edilmelidir. Sağa-sola dönüş için değer kolunda belirlenen miktarda 10 cm ilerlenmelidir. İleri-geri gitmek için değer kolunda belirtilen değer kadar ileri veya geri gidilmelidir.
6. İstenilen sayıda komut verilebilir.
7. Kuvvet sensörüne basıldığında kodlama işlemi bitecektir ve robot verilen komutları yerine getirecektir. Örneğin kol değeri 90 iken sola dön, 100 iken sağa dön, 20 iken ileri git, 30 iken geri git komutları sırası ile verildikten sonra kuvvet sensörüne basıldığında robot sırasıyla 10 cm boyunca -90 yönünde, 10 cm boyunca 100 yönünde, 20 cm boyunca ileri, 30 cm boyunca geri gitmelidir.

Bu görev için aşağıdaki Lego parçası kullanılabilir. Parçanın rengi sarı olmak zorunda değildir. Farklı renkteki parçalar da kullanılabilir. Komut verilirken Lego parçası kısa kenarı genişliği uzun kenarı ise yüksekliği olacak şekilde sensörlerin karşısında hareket etmelidir.



*Resim 10.8 Kullanılan Lego Parçası*

Sağa ve sola komutları için Lego parçası aşağıdaki resimde gösterildiği gibi sensörlere çok yakın olmalıdır. Komut vermek için başka Lego parçası da kullanılabilir. Fakat Lego parçasının mümkün mertebe büyük olması komut vermeyi kolaylaştıracaktır.



*Resim 10.9 Sola Dön Komutu*

Tanımlama: Bu aşamada öğrenciler problemi tanımlamalıdır. Problemi tanımlamak verilenleri, istenenleri, konu ile ilgili var olan bilgileri, ihtiyaçları ve geçmiş problem çözme deneyimlerini ortaya koymak açısından önemlidir. Bu bilgiler problem çözümünde kullanılabilir. Aşağıda örnek tanımlama adımları verilmiştir.

1. Sola dön ve sağa dön komutları iki aşamalı olarak düşünülebilir. Soldaki sensörden (renk) başlanıp sağdaki sensörün (mesafe) terk edilmesi sağa dön ve sağdaki sensörden başlanıp soldaki sensörün terk edilmesi ise sola dön anlamına gelebilir. Görüldüğü gibi komutlarda başlangıç ve terk etme gibi iki adım bulunmaktadır.
2. Sola veya sağa dönme komutları için Lego parçası sensörlerin çok yakından geçmelidir. Bu mesafe 6 cm olarak belirlenebilir.
3. İleri veya geri git komutları mesafe sensörü ile başlar ve mesafe sensöründe biter. Burada ikili bir yapı söz konusu değildir. Fakat genel bir programlama yapısı oluşturmak için bu komutlar da iki adımlı olarak düşünülebilir.
4. İleri git komutunda mesafe sensörüne yaklaşık 30 cm’den yaklaşmaya başlanmalı ve yaklaşık 15 cm mesafede komut bitirilmelidir.
5. Geri git komutunda mesafe sensöründen uzaklaşmaya yaklaşık 15 cm mesafeden başlanmalı ve 30 cm’de komut bitirilmelidir.
6. Verilecek komutlar arasına bir miktar süre konulması komutların daha rahat alınmasına yardımcı olabilir.
7. Dönme yönü ve ileri-geri gitme miktarı için büyük motordan elde edilecek değerler kullanılabilir.
8. Kuvvet sensörüne basılması komut verme işinin bittiği anlamına gelecektir. Kuvvet sensörüne basıldığında verilen komutlar yerine getirilmelidir.

Fikir Üretme: Bu aşamada öğrenciler problem çözümüne yönelik daha somut önerilerde bulunmalıdır. Hangi algoritmalar ve programlama yapıları kullanılarak problem çözülebilir, problem çözümü için gerekli parametreler nasıl seçilmelidir ve hangi programlama yöntemleri kullanılabilir gibi sorulara yanıt aranmaktadır. Aşağıda fikir üretme aşaması için bir örnek verilmiştir.

1. Sağa, sola, ileri ve geri komutlarının iki adımlı olarak planlanabilmesi için komutun başladığı ve bittiği evreler için iki farklı değişken kullanılabilir.
2. Komutlar ve değerler birer listede saklanabilir. Komutlar ve değerler aynı sırada alınarak çalıştırılmaları sıra bazlı gerçekleştirilebilir.
3. Print komutu kullanılarak büyük motordan alınan değerin konsolda görünmesi sağlanabilir.
4. Kol döndürülürken her bir derece için bir birimlik artış gerçekleştirilebilir. Değer 100’ü geçmemelidir. Geçerse program vasıtası ile düzeltilmelidir.
5. Soldan veya sağdan komut vermeye başlanması değişkenler yardımıyla belirlenebilir.
6. Başlangıç mesafesi 6 cm’den büyük ve 15 cm’den küçük olup bitişi 30 cm’den daha büyük olan komutlar ileri git olarak algılatılabilir.
7. Başlangıç mesafesi 30 cm’den büyük olup bitişi 15 cm ile 6 cm aralığında olan komutlar geri git olarak algılatılabilir.
8. Her bir komut arasındaki süre için 1.5 saniye belirlenebilir.
9. Kuvvet sensörü tıklandığında komut verme işleminin sonlandırılması için while döngüsü kullanılabilir.
10. For döngüsü ile komutlar ve değerleri sırayla alınıp uygun kodlar çalıştırılabilir.

Lego parçasıyla robot kodluyorum etkinliği görece zor bir etkinliktir. Öğrenciler tanımlama ve fikir üretme adımlarında çeşitli zorluklara karılaşıp bu adımları yerine getirmek istemeyebilirler. Bu gibi durumlarda öğrencilere problem çözmeye başlamadan önce problem üzerine düşünmenin faydalı olduğu ve uzman programcıların bu şekilde çalıştığı hatırlatılır. Öğrenciler tam olarak tanımlama ve fikir üretme adımlarını gerçekleştiremiyorsa, en azından doğrudan çözüme başlamak yerine öncelikle problem ve çözümü hakkında düşünmelidir. Öğrenciler her hâlükârda doğrudan probleme geçmek isterse, onlara çözüme başlamaları ve hızlı bir çözüm pratiğinin ardından yeniden genel bir problem çözüm tasarısı ortaya koymaları önerilmelidir.

## ÜRET

### Lego Parçasıyla Robotu Kodluyorum

Öğrenciler tasarlama adımında ortaya koyduğu temel üzerinden bilgisayar başında kod yazmaya başlarlar. Rehber öğretmen bu adımda öğreten pozisyonunda olmadığı için çözüme yönelik soruları doğrudan yanıtlamaz. Fakat problem görece zor olduğu için çözümün mantığı üzerine yorumlar yaparak öğrencileri yönlendirir. Bu etkinlik öğrencilere büyük bir problemi parçalara ayırarak çözme pratiğini göstermek için iyi bir örnektir. Öğrencilerden problemi bir bütün olarak çözmek yerine öncelikle sola dön komutu üzerine yoğunlaşmaları istenebilir. Bunu başardıktan sonra sırasıyla, sağa dön, ileri git, geri git ve büyük motordan değer oku gibi alt problemler üzerine yoğunlaşabilirler. Her bir alt çözümün doğruluğundan emin olduktan sonra, bu alt çözüm daha önce ortaya konan alt çözüme eklenebilir. Böylece problem çözümü alt çözümler eklenerek devam eder. Bunun yerine öğrenciler her bir alt problemi ayrı ayrı çözdükten sonra bu alt çözümleri birleştirme tasarısını yapabilir ve ardından bütün alt çözümleri birleştirerek problem çözümünü oluşturabilir. Aşağıda örnek bir kod verilmiştir. Öğrenciler bu koda benzer çözümler üretmek zorunda değildir, çok farklı çözümlere ulaşabilirler.

hub = PrimeHub()

mesafe\_sensoru=DistanceSensor("F")

renk\_sensoru=ColorSensor("B")

kuvvet\_sensoru=ForceSensor("A")

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

deger\_kolu=Motor("E")  # Büyük motora bağlı kol

deger\_kolu.run\_to\_position(0) # Kol başlangıçta 0'a getiriliyor

deger\_kolu.set\_degrees\_counted(0) # Açı değeri 0 olarak ayaranıyor

miktar\_yon=[] #Bu listede büyük motordan alınan değerler tutulacak

komutlar=[] #Bu listede ileri, geri, sag ve sol gibi komutlar tutulacak

basladi=False #Komut girmeye baslandi mi

bitti=True # Komuıt girme bitti mi

soldan\_giris=False # Soldan komut girmeye baslandi mi

sagdan\_giris=False # Sagdan komut girmeye baslandi mi

ileri\_git=False # İleri git komutu verildi mi

geri\_git=False # Geri git komutu verildi mi

mesafe=None # Mesafe sensöründen okunan deger

sayac=1 # Komut gir bip sesi sayacı, 1=bip çal; 2=bip çalma

while kuvvet\_sensoru.is\_pressed()==False: #Kuvvet sen basılana kadar komut gir

    deger=deger\_kolu.get\_degrees\_counted() # Deger\_kolu okunuyor

    if deger>100: #Deger move komutuna gideceği için 100'den büyük olamaz

        deger=100

    if deger<-100: #Deger move komutuna gideceği için -100'den küçük olamaz

        deger=-100

    print("deger:"+ str(deger)) #Girilen değer konsola basılıyor

    if basladi==False and bitti==True: # Deger girme baslangici

        if sayac==1: # Başlanıçta 1 kere bip çal

            hub.speaker.beep(60)

            sayac=2

        mesafe= mesafe\_sensoru.get\_distance\_cm()

        if  mesafe!=None and mesafe<20: # Bir cisim algılandıysa:

            if mesafe<6: # Çok yakın mesafe sagdan giriş demek

                sagdan\_giris=True

                basladi=True # Komut girmeye baslandi,

                bitti=False  #  ama henuz bitmedi.

            elif mesafe<15: # Yakından uzağa gitmek, geri git demek.

                mesafe\_sensoru.wait\_for\_distance\_farther\_than(30,"cm")

                ileri\_git=True

                basladi=True # Genel yapıya uyumlu olsun diye yapıldı,

                bitti=False # aslında komut girme burada bitti.

            elif mesafe<30: #Uzaktan yakına gelmek, geri git demek.

                mesafe\_sensoru.wait\_for\_distance\_closer\_than(15,"cm")

                geri\_git=True

                basladi=True # Genel yapıya uyumlu olsun diye yapıldı,

                bitti=False # aslında komut girme burada bitti.

            sayac=1

        if renk\_sensoru.get\_reflected\_light()>5: # Yansıyan ışık varsa:

            soldan\_giris=True # Komut girmeye soldan başlanmış

            basladi=True # Komut girmeye baslandi,

            bitti=False #ama henuz bitmedi.

            sayac=1

    if basladi==True and bitti==False: # Komutun devamı burada kontrol edilir

        if soldan\_giris==True: # Komut için soldan giriş yapıldısa:

            mesafe=mesafe\_sensoru.get\_distance\_cm()

            if mesafe !=None and mesafe<6: # sağdan çıkıldı mı?

                mesafe\_sensoru.wait\_for\_distance\_farther\_than(10,"cm")

                komutlar.append("saga\_don") # Sağa dön komutu verildi.

                miktar\_yon.append(deger) # Deger kaydedildi

                soldan\_giris=False # Yeniden değer girilebilmesi için

                hub.speaker.beep(100) # Komut girme bitti sesi

                wait\_for\_seconds(1.5) # Yeni komut için bekle

                basladi=False #Yeni komut girme henüz başlamadı

                bitti=True #Komut girme bitti

                deger\_kolu.run\_to\_position(0) # Değer kolunu yeni komut,

                deger\_kolu.set\_degrees\_counted(0) # için sıfırla.

        if sagdan\_giris==True: # Komut için sağdan giriş yapıldıysa:

            if renk\_sensoru.get\_reflected\_light()>5: #soldan çıkıldı mı?

                renk\_sensoru.wait\_for\_new\_color()

                komutlar.append("sola\_don") #Sola dön komutu verildi

                miktar\_yon.append(deger) # Deger kaydedildi

                sagdan\_giris=False # Yeniden değer girilebilmesi için

                hub.speaker.beep(100) # Komut girme bitti sesi

                wait\_for\_seconds(1.5) # Yeni komut için bekle

                basladi=False #Yeni komut girme henüz başlamadı

                bitti=True #Komut girme bitti

                deger\_kolu.run\_to\_position(0) # Değer kolunu yeni komut,

                deger\_kolu.set\_degrees\_counted(0) # için sıfırla.

        if ileri\_git==True: # İleri git komutu için:

            komutlar.append("ileri\_git") #komutu kaydet

            miktar\_yon.append(deger) #Değeri kaydet

            ileri\_git=False # Yeni komut girişi için

            hub.speaker.beep(100) # Komut girme bitti sesi

            wait\_for\_seconds(1.5) # Yeni komut için bekle

            basladi=False

            bitti=True

            deger\_kolu.run\_to\_position(0)

            deger\_kolu.set\_degrees\_counted(0)

        if geri\_git==True: # Geri git komutu için:

            komutlar.append("geri\_git") #komutu kaydet

            miktar\_yon.append(deger) #değeri kaydet

            geri\_git=False # Yeni komut girişi için

            hub.speaker.beep(100) # Komut girme bitti sesi

            wait\_for\_seconds(1.5)# Yeni komut için bekle

            basladi=False

            bitti=True

            deger\_kolu.run\_to\_position(0)

            deger\_kolu.set\_degrees\_counted(0)

for indeks in range(len(komutlar)): #Komutlar çalıştırılıyor

    if komutlar[indeks]=="ileri\_git":

        surus\_tabani.move(miktar\_yon[indeks],"cm",0)

    if komutlar[indeks]=="geri\_git":

        surus\_tabani.move(-miktar\_yon[indeks],"cm",0)

    if komutlar[indeks]=="saga\_don":

        surus\_tabani.move(10,"cm",miktar\_yon[indeks])

    if komutlar[indeks]=="sola\_don":

        surus\_tabani.move(10,"cm",-miktar\_yon[indeks])

## DEĞERLENDİR

Rehber öğretmen aşağıdaki soruları öğrencilere sorar ve sınıf tartışması içerisinde çözüm önerilerini ortaya koymalarını sağlar.

1. Bugün gerçekleştirilen programlama etkinliklerinden en zorlandığınız hangisidir? Sebebini açıklamaya çalışınız.
2. PID algoritması nedir? Genel olarak özetleyiniz.
3. PID günlük yaşamda nerelerde kullanılabilir?
4. Şimdiye kadar görülen bütün konular kullanılarak bir günlük yaşam problemi için çözüm önerisi getirmeniz beklenmektedir. Probleminizi tanımlayınız ve problem çözümünde kullanılacak robotik kavramlarını açıklayınız.

|  |
| --- |
| **PROJE HAZIRLIYORUM- FİKİR ÜRETME AŞAMASI** |
| Proje hazırlama süreçlerinde öğrenciler çevrelerinden veya günlük hayatlarından uygun bir problem belirleyip bu probleme kendi tasarladıkları bir robot ile çözüm üretirler. Çözümün başarılı olabilmesi için ihtiyaçların iyi tanımlanmış olması ve bu ihtiyaçların giderilmesi için uygun çözüm önerilerinin sunulması önemlidir. Bu hafta, empati ve tanımlama adımında toplanan bilgiler bir bütün hâline getirilir ve çözüm önerisi taslak olarak ortaya konur. Farklı çözümlerin önerilmesi gereklidir. Bunlar çok detaylı ve tamamlanmış fikirler olmak zorunda değildir fakat öneriler gerçekçi ve üretilmeye uygun olmalıdır.  Gruplardan belirledikleri robot projesi için yaptıkları fikir üretme çalışmasının sonucunu sınıfta açıklamaları istenir. Açıklamaların aşağıdaki bilgileri içermesi gerektiği belirtilir.   * Problem cümlesi: Robotun ne yapması isteniyor? * Robotun kullanılması planlanan ortamdaki ihtiyaçlar nelerdir? * Robotun kullanılması planlanan ortamdaki beklentiler nelerdir? * Hedeflerin gerçekleştirilmesi için yapılması gerekenler nelerdir? * Robotun belirlenen hedefleri gerçekleştirebilmesi için tasarımı nasıl olmalıdır? * Robotun belirlenen hedefleri gerçekleştirebilmesi için hangi işlemleri yapabilmesi gerekir? * Problem için üretilen alternatif tasarım ve programlama çözümleri nelerdir? * Tasarım ve programlama çözümünün seçim süreci nasıl gerçekleştirilmiştir? * Seçilen programlama çözümünün şematik veya maddeler hâlinde gösterimi (algoritması) nasıldır?   Açıklanan projelerle ilgili olarak diğer öğrencilerin ve rehber öğretmenlerin fikirleri alınır.  **Sonraki Haftaya Hazırlık:** Gruplara, üretilen fikirlere bağlı olarak robotun tasarlanıp programlanması aşamasına geçileceği söylenir. |