**Kodlar yapay zekadan yardım alınarak yazıldı. Bağlantı:**

**Donanım Bağlantıları**

**Motorlar ve ESC Bağlantıları**

| **Bileşen** | **Bağlantı Noktası** | **Açıklama** |
| --- | --- | --- |
| **Motor 1** | **ESC 1** | **Motor 1, ESC 1 üzerinden kontrol edilir.** |
| **Motor 2** | **ESC 2** | **Motor 2, ESC 2 üzerinden kontrol edilir.** |
| **ESC 1 Sinyal Kablosu** | **GPIO 14** | **ESC 1'in sinyal kablosu, Raspberry Pi'nin GPIO 18 pinine bağlanır (PWM).** |
| **ESC 2 Sinyal Kablosu** | **GPIO 15** | **ESC 2'in sinyal kablosu, Raspberry Pi'nin GPIO 19 pinine bağlanır (PWM).** |
| **ESC Güç Kabloları** | **Güç Dağıtım Kartı** | **ESC'lerin güç kabloları, güç dağıtım kartına bağlanır.** |
| **Güç Dağıtım Kartı** | **Lipo Batarya** | **Güç dağıtım kartı, Lipo bataryaya bağlanır.** |

**IMU Sensörü (BNO055) Bağlantıları**

| **BNO055 Pini** | **Raspberry Pi Pini** | **Açıklama** |
| --- | --- | --- |
| **SDA** | **GPIO 2 (SDA)** | **I2C veri hattı.** |
| **SCL** | **GPIO 3 (SCL)** | **I2C saat hattı.** |
| **GND** | **GND** | **Toprak (Ground) bağlantısı.** |
| **VCC** | **3.3V** | **Sensörün güç beslemesi (3.3V).** |

**Raspberry Pi Bağlantıları**

| **Bileşen** | **Bağlantı Noktası** | **Açıklama** |
| --- | --- | --- |
| **Raspberry Pi Güç** | **Lipo Batarya veya Harici Kaynak** | **Raspberry Pi, Lipo bataryadan veya harici bir güç kaynağından beslenir.** |
| **ESC Sinyal Kabloları** | **GPIO 18 ve GPIO 19** | **ESC'lerin sinyal kabloları, Raspberry Pi'nin GPIO pinlerine bağlanır.** |

**kp, ki ve kd parametreler şu şekilde ayarlanmalı:  
Başlangıçta kp'yi küçük bir değerle başlatın (örneğin, kp = 1.0).**

* **Sistemin tepkisini gözlemleyin:**
  + **Eğer sistem yavaşsa, kp'yi artırın.**
  + **Eğer aşırı salınımlar varsa, kp'yi azaltın.**

**Sonra ki'yi çok küçük bir değerle başlatın (örneğin, ki = 0.01).**

* **Sistemin tepkisini gözlemleyin:**
  + **Eğer sistem hedef değere ulaşamıyorsa, ki'yi artırın.**
  + **Eğer aşırı salınımlar varsa, ki'yi azaltın.**

**Son olarak kd'yi küçük bir değerle başlatın (örneğin, kd = 0.1).**

* **Sistemin tepkisini gözlemleyin:**
  + **Eğer aşırı salınımlar varsa, kd'yi artırın.**
  + **Eğer sistem titriyorsa, kd'yi azaltın.**

**Gözlemler:**

1. **Sistem yavaş tepki veriyorsa:**
   * **kp'yi artırın (örneğin, kp = 2.0).**

**Motorlar aşırı salınım yapıyorsa (oscilasyon):**

* + **Kp değerini azalt.**

1. **Sistem hedef değere ulaşamıyorsa:**
   * **ki'yi artırın (örneğin, ki = 0.02).**

**Motorlar aşırı salınım yapıyorsa:**

* + **Ki değerini azalt.**

1. **Aşırı salınımlar varsa:**
   * **kd'yi artırın (örneğin, kd = 0.2).**

**Motorlar çok yavaş tepki veriyorsa:**

* + **Kd değerini azalt.**

**PID (Proportional-Integral-Derivative) kontrolörünün** temel bileşenleridir. PID kontrolörü, bir sistemin çıktısını belirli bir hedef değere (setpoint) getirmek ve bu değerde tutmak için kullanılan bir geri besleme kontrol mekanizmasıdır. Bu değişkenler, PID kontrolörünün hesaplamalarını yapmak için kullanılır. Şimdi her bir değişkenin ne işe yaradığını detaylıca açıklayalım:

**1. self.kp (Proportional Gain - Oransal Kazanç)**

* **Ne İşe Yarar?**
  + PID kontrolörünün "P" (Proportional) kısmını temsil eder.
  + Sistemin mevcut hatası (error) ile doğrudan orantılı bir çıktı üretir.
  + Hata ne kadar büyükse, kontrol çıktısı da o kadar büyük olur.
* **Matematiksel İfade:**

Poutput=Kp×error*P*output​=*Kp*​×error

* **Örnek:**
  + Eğer error = 10 ve K\_p = 2 ise, P\_output = 20 olur.
* **Etkisi:**
  + K\_p büyükse sistem hızlı tepki verir, ancak aşırı salınımlara (overshoot) veya kararsızlığa neden olabilir.
  + K\_p küçükse sistem yavaş tepki verir ve hedef değere ulaşması uzun sürer.

**2. self.ki (Integral Gain - İntegral Kazanç)**

* **Ne İşe Yarar?**
  + PID kontrolörünün "I" (Integral) kısmını temsil eder.
  + Sistemin geçmişteki hatalarının toplamını (integralini) kullanarak kalıcı hataları (steady-state error) ortadan kaldırır.
* **Matematiksel İfade:**

Ioutput=Ki×∫error dt*I*output​=*Ki*​×∫error*dt*

* + Burada self.integral, hataların toplamını temsil eder.
* **Örnek:**
  + Eğer error = [2, 3, 1] ve K\_i = 0.1 ise, I\_output = 0.1 \* (2 + 3 + 1) = 0.6 olur.
* **Etkisi:**
  + K\_i büyükse sistem kalıcı hataları hızlıca düzeltir, ancak aşırı salınımlara neden olabilir.
  + K\_i küçükse kalıcı hatalar yavaşça düzeltilir.

**3. self.kd (Derivative Gain - Türevsel Kazanç)**

* **Ne İşe Yarar?**
  + PID kontrolörünün "D" (Derivative) kısmını temsil eder.
  + Sistemin hatasındaki değişim hızını (türev) kullanarak aşırı salınımları (overshoot) ve kararsızlığı önler.
* **Matematiksel İfade:**

Doutput=Kd×d(error)dt*D*output​=*Kd*​×*dtd*(error)​

* + Burada self.prev\_error, bir önceki hata değerini temsil eder.
* **Örnek:**
  + Eğer error = 10, prev\_error = 8 ve K\_d = 0.5 ise, D\_output = 0.5 \* (10 - 8) = 1 olur.
* **Etkisi:**
  + K\_d büyükse sistem hızlı değişimlere karşı daha duyarlı olur, ancak gürültüye karşı hassas hale gelebilir.
  + K\_d küçükse sistem yavaş tepki verir ve salınımlar artabilir.

**4. self.prev\_error (Önceki Hata)**

* **Ne İşe Yarar?**
  + Bir önceki adımdaki hata değerini saklar.
  + Türevsel (D) kısmın hesaplanması için kullanılır.
* **Örnek:**
  + Eğer bir önceki hata prev\_error = 5 ve şu anki hata error = 8 ise, türevsel çıktı:

Doutput=Kd×(8−5)=3Kd*D*output​=*Kd*​×(8−5)=3*Kd*​

**5. self.integral (Hataların Toplamı)**

* **Ne İşe Yarar?**
  + Geçmişteki tüm hataların toplamını saklar.
  + İntegral (I) kısmının hesaplanması için kullanılır.
* **Örnek:**
  + Eğer hatalar [2, 3, 1] ise, integral = 2 + 3 + 1 = 6 olur.
  + İntegral çıktısı:

Ioutput=Ki×6*I*output​=*Ki*​×6

**PID Kontrolörünün Genel Çalışma Prensibi**

PID kontrolörü, yukarıdaki üç bileşeni (P, I, D) birleştirerek bir çıktı üretir:

Output=Poutput+Ioutput+DoutputOutput=*P*output​+*I*output​+*D*output​

* **P (Proportional):** Mevcut hataya tepki verir.
* **I (Integral):** Geçmiş hataları düzeltir.
* **D (Derivative):** Gelecekteki hataları önler.

**Örnek Senaryo**

Aracınızın **roll** açısını dengede tutmak için PID kontrolörü kullanıyorsunuz:

1. **Hedef (Setpoint):** target\_roll = 0 (dengede olma durumu).
2. **Ölçülen Değer (Measured Value):** current\_roll = 5 (sensor verisi).
3. **Hata (Error):** error = target\_roll - current\_roll = -5.

PID kontrolörü bu hatayı alır ve:

* **P:** K\_p \* (-5)
* **I:** K\_i \* (integral)
* **D:** K\_d \* (error - prev\_error)

Bu değerler toplanarak motor hızları ayarlanır ve araç dengede tutulur.

**Sonuç**

* self.kp, self.ki, self.kd: PID kontrolörünün kazanç değerleridir. Sistemin davranışını belirler.
* self.prev\_error: Türevsel (D) kısmın hesaplanması için kullanılır.
* self.integral: İntegral (I) kısmın hesaplanması için kullanılır.