Documentación del Código

Team mustabot

August 4, 2023

1 Clase `MOTOR`

Esta clase representa un motor en el robot. Proporciona métodos para controlar la velocidad del motor y obtener su nombre.

```
class MOTOR:
    def __init__(self, robot, name):
        # Constructor
        ...

    def setVelocity(self, velocity):
        # Establece la velocidad del motor
        ...

    def getVelocity(self):
        # Devuelve la velocidad actual del motor
        ...

    def getName(self):
        # Devuelve el nombre del motor
        ...

    def stop(self):
        # Detiene el motor
        ...
```

2 Clase `ENCODER`

Esta clase representa un encoder en el robot. Proporciona métodos para leer el valor del encoder, establecer su posición inicial y obtener su nombre.

```
class ENCODER:
    def __init__(self, robot, name):
        # Constructor
```

```
        ...

    def getValue(self):
        # Devuelve el valor actual del codificador
        ...

    def getName(self):
        # Devuelve el nombre del codificador
        ...

    def set_start_position(self):
        # Establece la posicion inicial del codificador
        ...
```

3 Funciones `doblar_derecha` y `doblar_izquierda`

Estas funciones calculan el tiempo y la potencia necesarios para que el robot gire hacia la derecha o hacia la izquierda en una fracción de una vuelta completa. La fracción se determina mediante el valor de entrada a la función.

```
def doblar_derecha(fraccion):
    # Calcula el tiempo y la potencia para girar a la
    # derecha
    ...

def doblar_izquierda(fraccion):
    # Calcula el tiempo y la potencia para girar a la
    # izquierda
    ...
```

4 Función `normalize_values`

Esta función ajusta los valores de los sensores para mejorar el rendimiento del controlador PID cuando se enfrenta a colores como el verde.

```
def normalize_values(values):
    # Ajusta los valores de los sensores
    ...
```

5 Función `PID_LINE`

Esta función implementa un controlador PID para seguir una línea utilizando lecturas de sensores. Calcula el error, ajusta la velocidad de los motores en

consecuencia y devuelve las velocidades actualizadas para el motor 1 y el motor 2.

```
def PID_LINE(kp, kd, ki, previous_error, sensor_values
    PBASE, sum_error, motor1, motor2):
    # Implementa el controlador PID para seguir la linea
    ...
```

6 Clases IR, LASER y COLOR

Estas clases representan el sensor infrarrojo, el sensor láser y el sensor de color, respectivamente. Proporcionan métodos para leer los valores de los sensores y obtener los nombres de los sensores.

```
class IR:
    def __init__(self, robot, name):
        # Constructor
        ...

    def getValue(self):
        # Devuelve el valor actual del sensor infrarrojo
        ...

    def getName(self):
        # Devuelve el nombre del sensor infrarrojo
        ...

class LASER:
    def __init__(self, robot, name):
        # Constructor
        ...

    def getValue(self):
        # Devuelve el valor actual del sensor laser
        ...

    def getName(self):
        # Devuelve el nombre del sensor laser
        ...

class COLOR:
    def __init__(self, robot, name):
        # Constructor
        ...
```

```
    def getValue(self):
        # Devuelve el valor actual del sensor de color
        ...

    def getName(self):
        # Devuelve el nombre del sensor de color
        ...

    def get_color(self):
        # Devuelve el color detectado por el sensor
        ...
```