Módulo PID

Objeto PID

Para crear un objeto PID, se utiliza la siguiente sintaxis:

PIDPIDMotor(Kp, Ki, Kd, St, MAX_OUT, T_error);

Donde:

- Kp: Ganancia proporcional
- Ki: Ganancia integral
- Kd: Ganancia derivativa
- sampling_time: Tiempo de muestreo en microsegundos
- max_out: Valor máximo de salida del PID
- error_tolerance: Tolerancia de error para considerar el sistema estable

Cálculo de la Salida

Existen dos funciones para calcular la salida del PID:

- **Control PID con Límite Integral:**
  ```
  // Establecer un set_point
  set_point = 700;
  // Obtener la posición actual del motor
  input = motor.getPulses();
  // Calcular la salida con límite integral
  output = PIDMotor.compute(input, set_point);
  // Aplicar la salida al motor
  motor.turn(output);
  ```

- **Control PID Normal:**
  ```
  // Establecer un set_point
  set_point = 700;
  // Obtener la posición actual del motor
  input = motor.getPulses();
  // Calcular la salida
  output = PIDMotor.compute_2(input, set_point);
  // Aplicar la salida al motor
  motor.turn(output);
  ```

Módulo para Motor

Objeto MotorEncoder

Para crear un objeto MotorEncoder, se utiliza la siguiente sintaxis:

MotorEncoder motor(PinM1, PinM2, PinPWM, A, B, PPG);

Donde:

- PinM1, PinM2: Pines de control de dirección del motor
- PinPWM: Pin para control de velocidad (PWM)
- PinA, PinB: Pines del encoder
- PPG: Pulsos por revolución del encoder (pul_Rv)

Funciones del Motor

- **motor.init():** Esta función se debe llamar siempre en la función void setup() y es esencial para la inicialización del objeto motor.
- **motor.configureESP32PWM(canal, frecuencia, resolución):** Configura la frecuencia y la resolución del PWM para el motor.
  - canal: El canal PWM a utilizar.
  - frecuencia: La frecuencia del PWM en Hz.
  - resolución: El número de bits de resolución del PWM (0 a 255 en este caso).
- **motor.setEncoderFilter_T(Filter):** Establece un filtro para reducir el ruido de la señal del encoder. El valor del filtro es el período de la señal cuadrada (T) medida con un osciloscopio, en microsegundos.
- **motor.Stop():** Detiene el motor.
- **motor.turn(output):** Controla la velocidad y dirección del motor.
  - output = 255: Máxima velocidad en un sentido.
  - output = -255: Máxima velocidad en el sentido opuesto.
- **motor.getSpeed():** Devuelve la velocidad del motor en RPM (revoluciones por minuto) como un valor de tipo float.
- **motor.getDegrees():** Devuelve la posición angular del motor en grados como un valor de tipo float.
- **motor.getPulses():** Devuelve el número total de pulsos del encoder como un valor de tipo long.