**NGHIÊN CỨU KHOA HỌC**

<https://drive.google.com/drive/u/0/folders/1ZZXsrGXiXf2hmRYYXqTpE0Vwej-OF110> (tài liệu vi xử lý)

Ý tưởng:

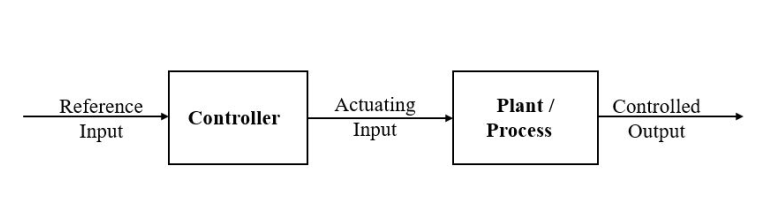
- Sử dụng cảm biến để giúp xe phát hiện và theo dõi đường line. Cảm biến có thể là cảm biến màu sắc, cảm biến hồng ngoại, hoặc các loại cảm biến quang điện.

- PID Controller (Proportional, Integral, Derivative): để duy trì và điều khiển vị trí của xe

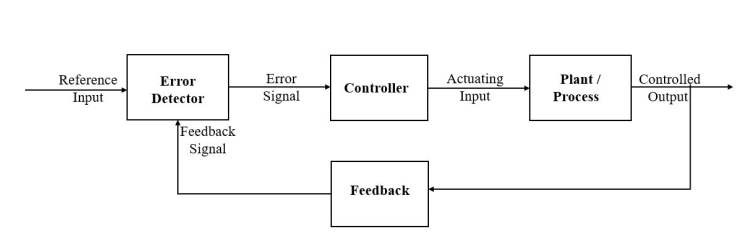
Lý thuyết PID Controller

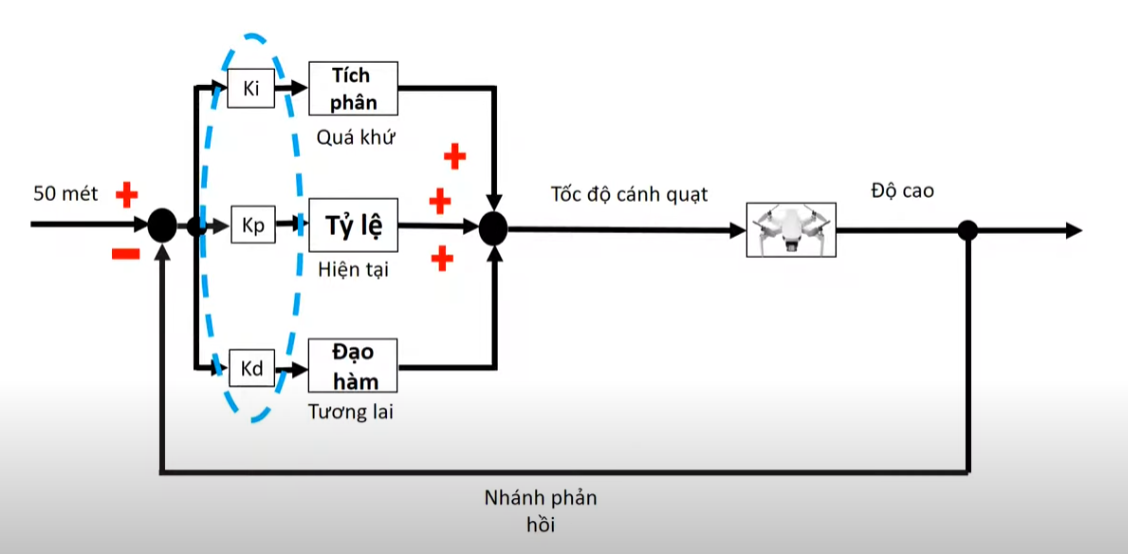
Open loop control system

Plant/Process: Hệ thống mà chúng ta muốn điều khiển



Close loop control system

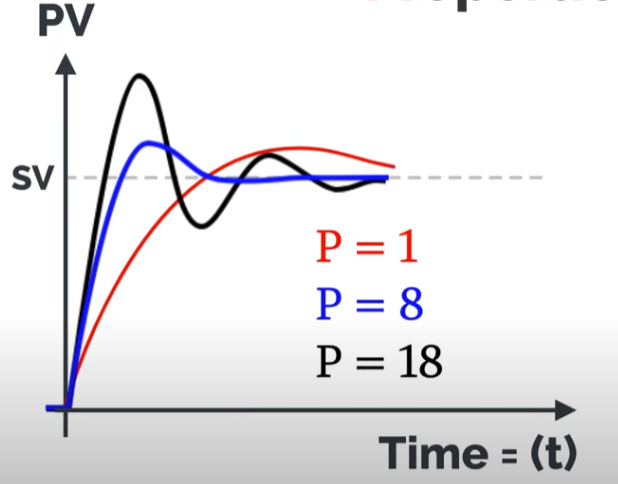
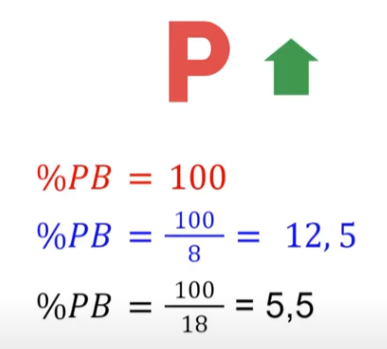




Bộ điều khiển PID bao gồm ba thành phần chính:

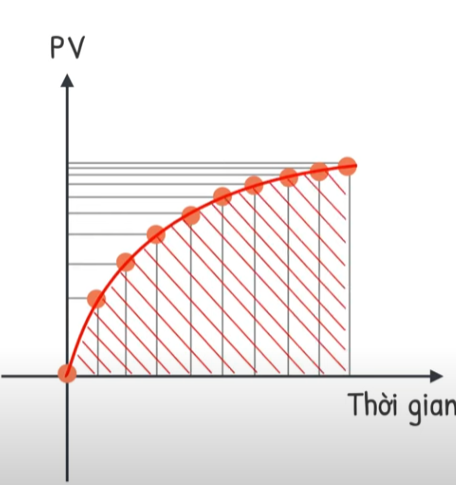
1. **Thành phần Proportional (P):**

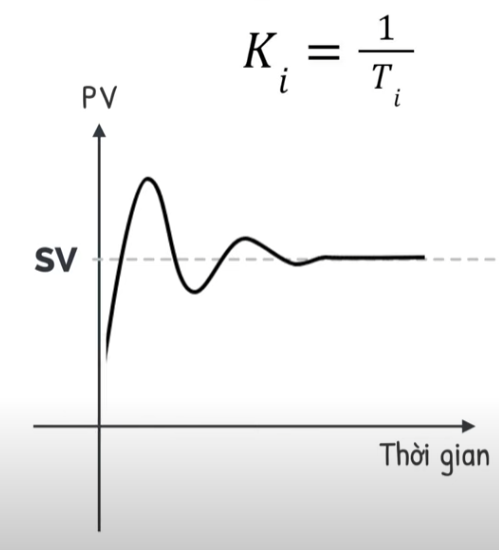
* Thành phần này **tỉ lệ** với sự sai khác giữa giá trị đầu vào hiện tại và giá trị setpoint. Nó tịa ra một tín hiệu điều khiển tỉ lệ với sai số, và nó giúp tiếp cận setpoint nhanh hơn.
* Là thông số quyết định tốc độ mà hệ thống sẽ phản hồi lại với sự thay đổi trong quá trình điều khiển tự động.

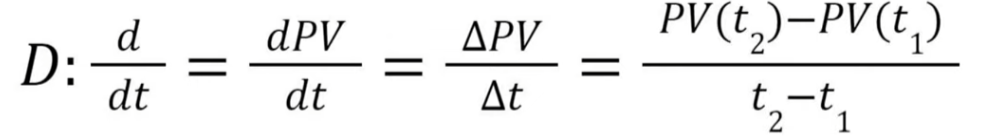
* Việc tăng thông số điều chỉnh này có thể gây ra các vòng lặp điều khiển nhạy hơn nhưng kém ổn định hơn, và ngược lại (nó có thể đẫn đến hiện tượng dao động (oscillation) nếu chỉ sử dụng thành phần này).

1. **Thành phần Integral (I):** Thành phần này **tích lũy** và **điều chỉnh** sai số tích lũy trong thời gian. Nó giúp khắc phục sai số ổn định và đưa hệ thống về setpoitnt mà không bị sai số dài hạn. Thành phần này giúp loại bỏ hiện tượng ofset, nhưng nếu được cài đặt quá cao, nó có thể dẫn đến hiện tượng overshoot.

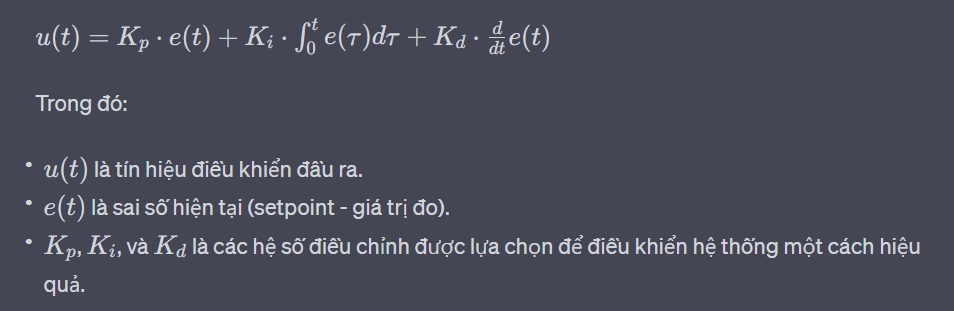
 Tổng tất cả các giá trị từ khi bắt đầu đếm đến khi hoàn thành việc đếm (diện tích dưới đường cong được vẽ)



1. **Thành phần Derivate (D):** Thành phần này ước tính sự thay đổi của sai số theo thời gian. Nó giúp **dự đoán** sự biển đổi của hệ thống và **giảm độ dao dộng** khi hệ thống tiếp cận setpoint. Thành phần này giúp ổn định hệ thống và ngăn chặn hiện tượng overshoot.

* **Hằng số D** :  Phản ánh tốc độ biến quy trình

**Các thành phần PID được kết hợp lại để tạo ra một tín hiệu điều khiển tổng hợp, được tính bằng cách cộng tổng của các thành phần P, I và D.** Công thức tổng hợp thường có dạng như sau:



* e (sai số) = SP (setpoint) – PV (processvariable)

Ví dụ: Thuật toán điều khiển xe chạy trên một đường thẳng bằng PID có thể được thực hiện theo cách sau:

* Kp (Proportional Band): %PB = 100/P ( được gọi là tỷ lệ /độ lợi)

1. **Cài đặt cảm biến**: Đầu tiên, bạn cần một cảm biến để đo vị trí của xe trên đường thẳng. Điều này có thể được thực hiện bằng cách sử dụng một cảm biến quang học, hệ thống GPS, hoặc bất kỳ phương pháp nào khác phù hợp.
2. **Thiết lập PID**: Xác định các hệ số *Kp*​, *Ki*​, và *Kd*​ cho PID controller. Các giá trị này cần được hiệu chỉnh dựa trên đặc điểm cụ thể của hệ thống và đường thẳng mà bạn muốn xe điều khiển trên đó.
3. **Thu thập dữ liệu và phản hồi**: Đọc giá trị từ cảm biến vị trí và tính toán sai số (error) giữa vị trí hiện tại của xe và vị trí mục tiêu trên đường thẳng. Sai số này là *e*(*t*) trong công thức PID.
4. **Tính toán tín hiệu điều khiển**: Sử dụng công thức PID để tính toán tín hiệu điều khiển dựa trên sai số *e*(*t*), sai số tích lũy từ quá khứ và tỷ lệ của thay đổi sai số.
5. **Điều khiển xe**: Sử dụng tín hiệu điều khiển tính được từ PID để điều khiển động cơ hoặc bánh xe của xe, sao cho xe điều chỉnh vị trí của mình để tiến tới vị trí mục tiêu trên đường thẳng. Thường, nếu tín hiệu điều khiển là dương, xe sẽ di chuyển về phía trước, và nếu là âm, xe sẽ di chuyển về phía sau.
6. **Lặp lại quá trình**: Lặp lại các bước trên liên tục để duy trì xe trên 1 đường thẳng*.*

* **Lập trình PID :**
  + **Setpoint**(giá trị mục tiêu)**:** vị trí trung tâm của đường line mà xe cố gắng duy trì.
  + **Output** (đầu ra PID) : Điều khiển động cơ lái hoặc các bánh xe để điều chỉnh hướng duy chuyển của xe
  + **Thành phần P,I, và D**: Điều chỉnh để đảm bảo xe có thể duy trì vị trí trên đường line một cách chính xác và ổn định

***Code ví dụ về cài đặt xe chạy trên 1 line****:*

**#include <Arduino.h>**

**// Khai báo chân kết nối với encoder và động cơ**

const int encoderPinA = 2;

const int encoderPinB = 3;

const int motorPin = 9;

**// Khai báo các hệ số PID**

double kp = 0.1;

double ki = 0.01;

double kd = 0.05;

**// Khai báo biến PID**

double setpoint = 100.0; **// Vị trí mục tiêu trên đường thẳng**

double current\_position = 0.0;

double prev\_position = 0.0;

double integral = 0.0;

double derivative = 0.0;

double output = 0.0;

**void setup()** {

pinMode(encoderPinA, INPUT);

pinMode(encoderPinB, INPUT);

pinMode(motorPin, OUTPUT);

attachInterrupt(digitalPinToInterrupt(encoderPinA), updatePosition, CHANGE);

}

**void loop()** {

**// Tính thời gian lặp**

unsigned long currentTime = millis();

static unsigned long prevTime = currentTime;

double dt = (double)(currentTime - prevTime) / 1000.0;

**// Tính sai số vị trí**

double error = setpoint - current\_position;

**// Tính các thành phần PID**

integral += error \* dt;

derivative = (current\_position - prev\_position) / dt;

output = kp \* error + ki \* integral + kd \* derivative;

**// Điều khiển động cơ**

analogWrite(motorPin, output);

**// Lưu lại vị trí hiện tại**

prev\_position = current\_position;

**// Cập nhật thời gian lặp**

prevTime = currentTime;

}

**void updatePosition()** {

**// Đọc giá trị encoder và cập nhật vị trí hiện tại**

int encoderStateA = digitalRead(encoderPinA);

int encoderStateB = digitalRead(encoderPinB);

if (encoderStateA == HIGH && encoderStateB == LOW) {

current\_position++;

} else if (encoderStateA == LOW && encoderStateB == HIGH) {

current\_position--;

}

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

- Hướng điểm tới (Heading Control): Sử dụng cảm biến la bàn hoặc cảm biến gia tốc-giá quay để theo dõi hướng di chuyển và điều chỉnh hướng của xe. Thuật toán đảm bảo xe duy trì một hướng hoặc góc xác định với mục tiêu hoặc hướng di chuyển mong muốn – điều khiển một xe điểm đến một đích cụ thể hoặc theo một đường thẳng cố định.

**class PIDController:**

**def \_\_init\_\_(self, kp, ki, kd):**

self.kp = kp

self.ki = ki

self.kd = kd

self.prev\_error = 0

self.integral = 0

**def compute(self, target\_heading, current\_heading):**

error = target\_heading - current\_heading

self.integral += error

derivative = error - self.prev\_error

control\_output = self.kp \* error + self.ki \* self.integral + self.kd \* derivative

self.prev\_error = error

return control\_output

**if \_\_name\_\_ == "\_\_main\_\_":**

pid = PIDController(kp=0.1, ki=0.01, kd=0.05)

target\_heading = 90 # Hướng mục tiêu (độ)

current\_heading = 0 # Hướng hiện tại (độ)

for \_ in range(100):

control\_signal = pid.compute(target\_heading, current\_heading)

# Áp dụng tín hiệu điều khiển (chẳng hạn, điều khiển bánh xe) để thay đổi hướng

current\_heading += control\_signal

print(f"Current Heading: {current\_heading:.2f} degrees")

\* Thuật toán Fuzzy Logic:

<https://www.researchgate.net/publication/235986365_Design_of_Fuzzy_Logic_Controller_for_Autonomous_parking_of_Mobile_Robot> (tài liệu về xe tự động tìm chổ đậu dùng fuzzy logic)

2. Thuật toán xử lý dữ liệu từ cảm biến:

- Fusion Sensor: Sử dụng các thuật toán như Bộ lọc Kalman (Kalman Filter) hoặc Bộ lọc hợp nhất (Sensor Fusion) để kết hợp dữ liệu từ nhiều cảm biến như cảm biến siêu âm, cảm biến hồng ngoại và cảm biến tiệm cận để xác định vị trí, hướng và tránh vật cản.:

<https://www.youtube.com/watch?v=6qV3YjFppuc>

<https://www.youtube.com/watch?v=0rlvvYgmTvI&t=293s>

<https://www.youtube.com/watch?v=hN8dL55rP5I>

- Thuật toán phát hiện và tránh vật cản: Áp dụng các thuật toán như giải thuật A\* (A-star), giải thuật RRT (Rapidly-exploring Random Trees) hoặc các thuật toán dựa trên máy học để phát hiện và tránh vật cản trong quá trình di chuyển của xe.

3. Thuật toán học máy:

- Học tăng cường (Reinforcement Learning): Sử dụng thuật toán học tăng cường như Q-Learning hoặc Deep Q-Network (DQN) để cho xe tự học và tối ưu hóa hành vi di chuyển dựa trên phản hồi từ môi trường xung quanh.

- Mạng nơ-ron nhân tạo (Artificial Neural Networks): Sử dụng mạng nơ-ron nhân tạo để xây dựng mô hình dự đoán và điều khiển xe dựa trên dữ liệu từ cảm biến và các thông số đầu vào.