PID Controller

Nguyễn Tiến Đạt -

Tổng quan

- PID Controller:

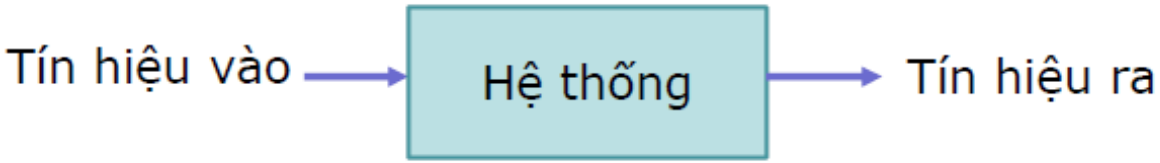
Wikipedia

Bộ điều khiển PID là một cơ chế vòng điều khiển sử dụng phản hồi (vòng kín), được sử dụng rộng rãi trong các hệ thống điều khiển công nghiệp và nhiều ứng dụng khác đòi hỏi kiểm soát liên tục được điều chỉnh.

Nội dung

1. Hệ thống và nhận dạng hệ thống
2. Cầu H và Encoder
3. Bộ điều khiển vòng kín và bộ điều khiển PID
4. Thiết kế bộ điều khiển PID cho động cơ DC
5. Phụ lục

1. Hệ thống và nhận dạng hệ thống



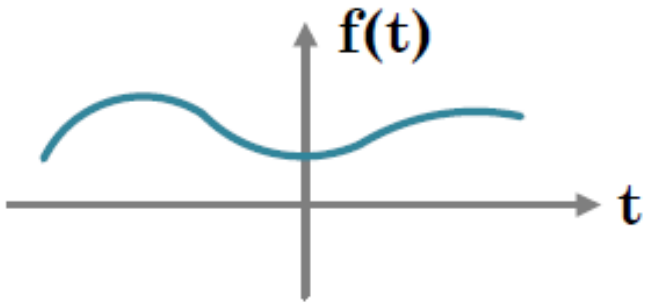
Tín hiệu: hàm của một hoặc nhiều biến độc lập (thời gian, không gian,...) mang thông tin về hành vi hoặc bản chất của các hiện tượng.

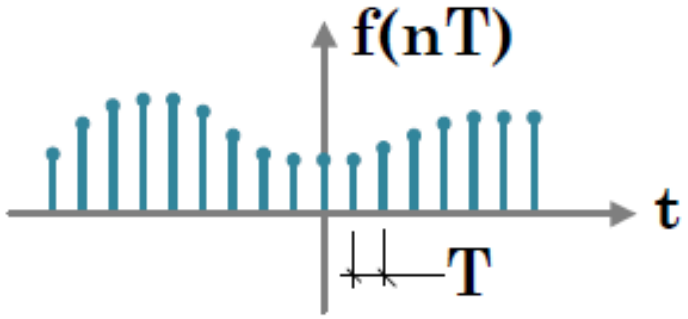
Hệ thống: mô hình vật lý hoặc giải thuật để xử lý tín hiệu vào và tạo tín hiệu ra.

1. Hệ thống và nhận dạng hệ thống

Tín hiệu liên tục: tín hiệu có giá trị tại mọi thời điểm (liên tục trên thang thời gian)

Tín hiệu rời rạc: tín hiệu có giá trị tại các thời điểm xác định (rời rạc trên thang thời gian)

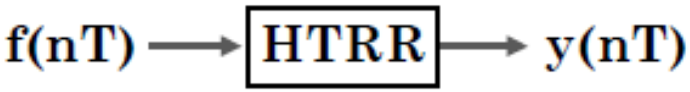




1. Hệ thống và nhận dạng hệ thống

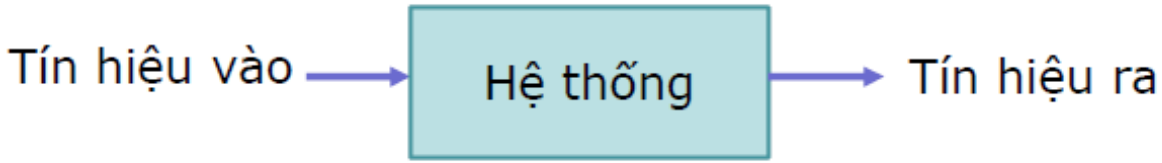
Hệ thống liên tục: tín hiệu vào liên tục, tín hiệu ra liên tục

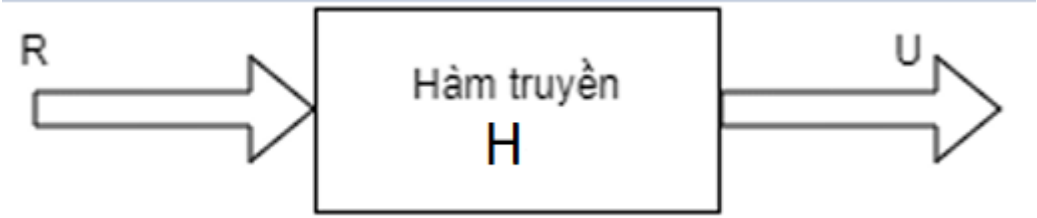
Hệ thống rời rạc: tín hiệu vào rời rạc, tín hiệu ra rời rạc



1. Hệ thống và nhận dạng hệ thống

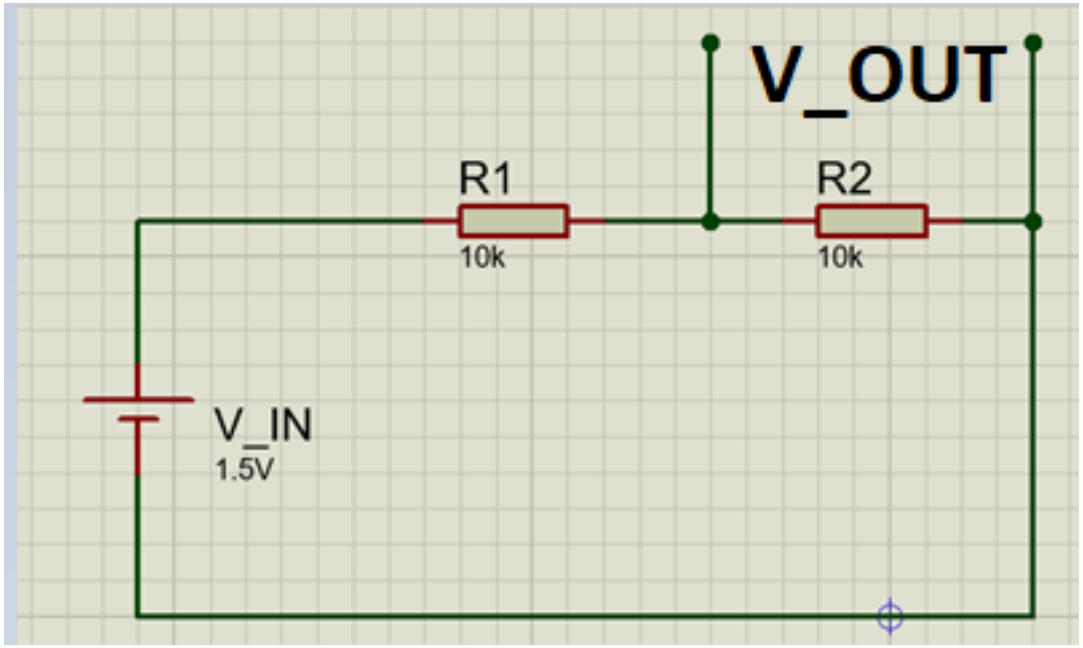
- Hàm truyền là một hàm số mô tả mối quan hệ giữa tín hiệu vào và tín hiệu ra của một hệ thống điều khiển.





1. Hệ thống và nhận dạng hệ thống

Ví dụ 1.1



1. Hệ thống và nhận dạng hệ thống

Ví dụ 1.1

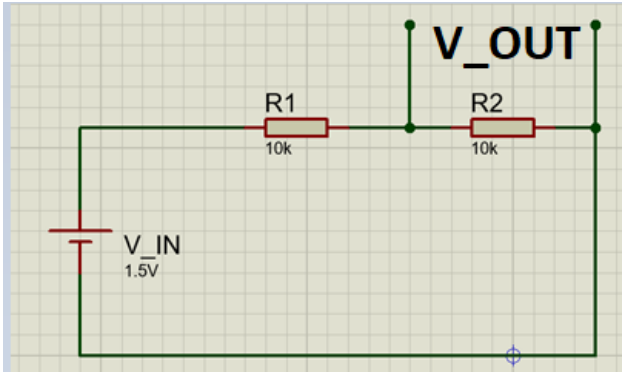
Hệ thống liên tục

Tín hiệu vào: V_IN

Tín hiệu ra: V_OUT

Quan hệ giữa tín hiệu vào/ra: V_OUT = 1/2 V_IN

Hàm truyền H = V_OUT / V_IN = 1/2



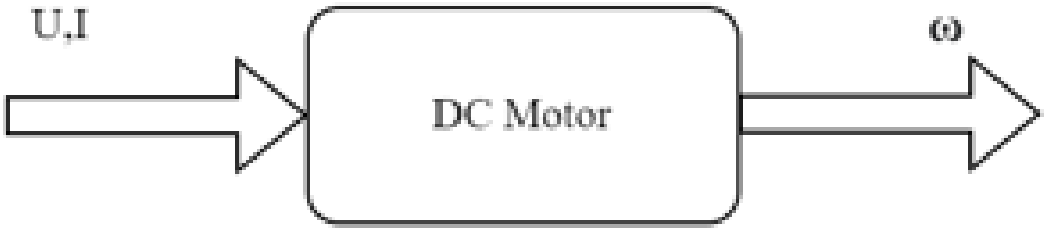
1. Hệ thống và nhận dạng hệ thống

Động cơ DC:
- Tín hiệu vào
- Tín hiệu ra
- Quan hệ vào/ra

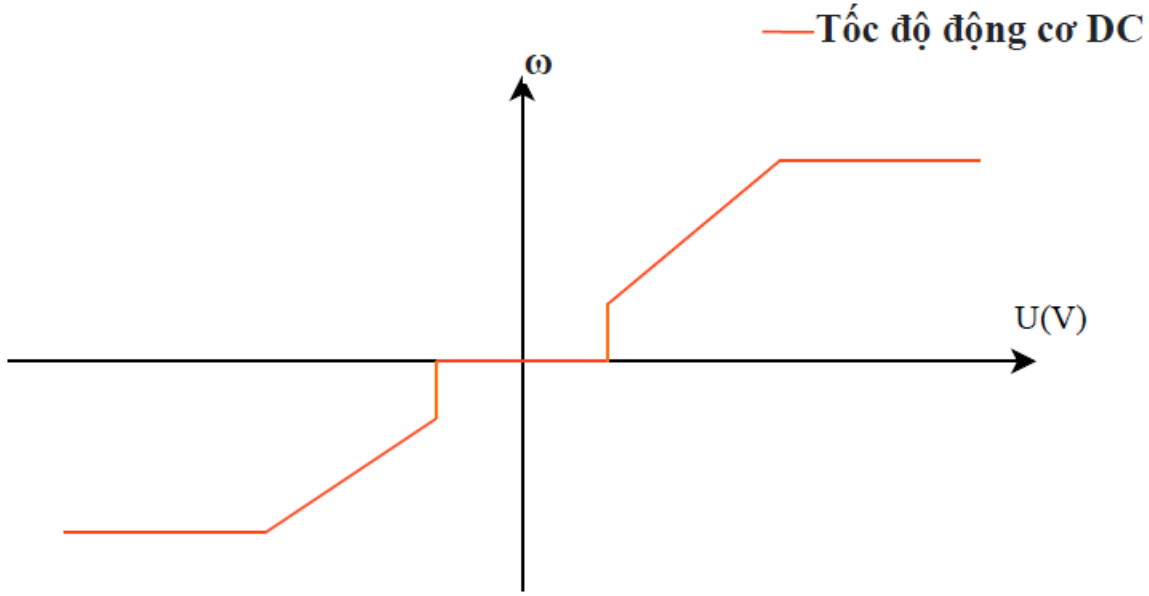
1. Hệ thống và nhận dạng hệ thống

- Động cơ DC:

- Tín hiệu vào: Điện áp U, dòng điện I.
- Tín hiệu ra: Tốc độ quay **ω** .
- Quan hệ vào/ra:

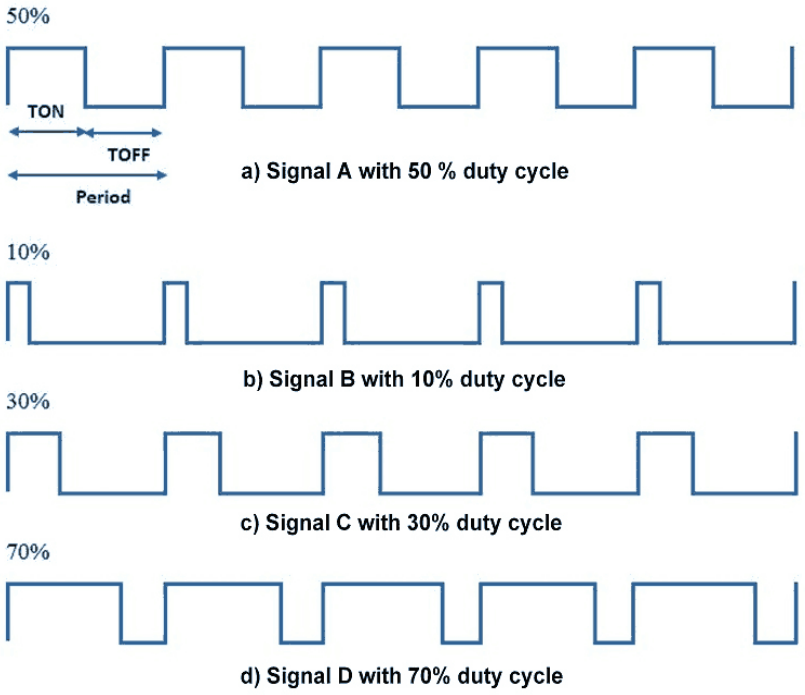


1. Hệ thống và nhận dạng hệ thống



2. Cầu H và Encoder

- Pulse Width Modulation

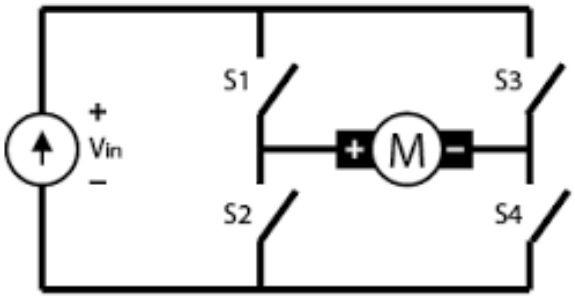


- Để điều khiển động cơ theo mong muốn thì ta sẽ thay đổi điện áp cấp vào động cơ.
- Phương pháp phù hợp nhất là PWM (Pulse Width Modulation).

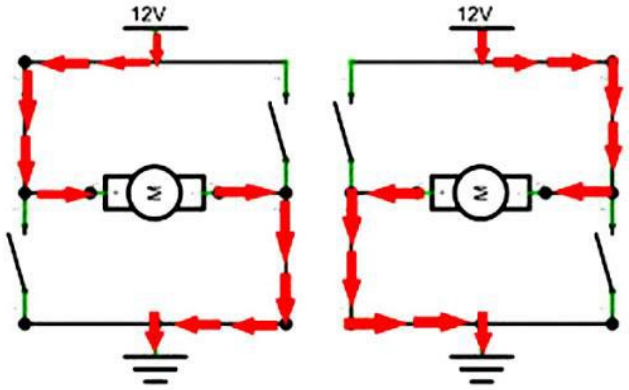
$$U_t = U * \frac{T_{on}}{T}$$

2. Cầu H và Encoder

- Cầu H:



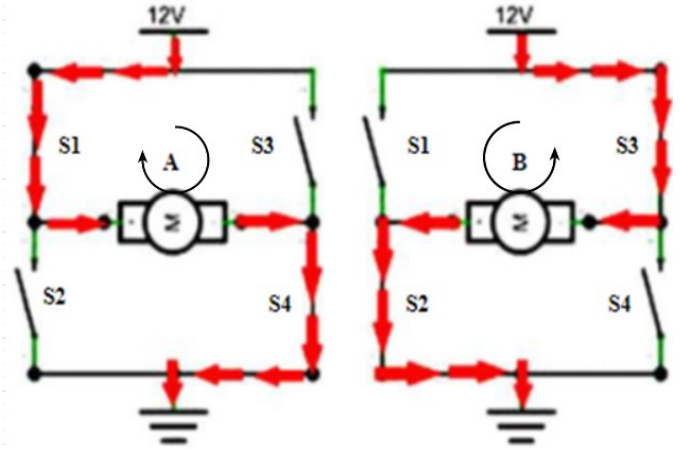
Mạch cầu H bao gồm 4 khóa điều khiển, được ký hiệu là S1, S2, S3 và S4.
Hai khóa (S1 và S4) được sử dụng để kiểm soát động cơ theo chiều một, trong khi hai khóa còn lại (S2 và S3) kiểm soát động cơ theo chiều ngược lại.



2. Cầu H và Encoder

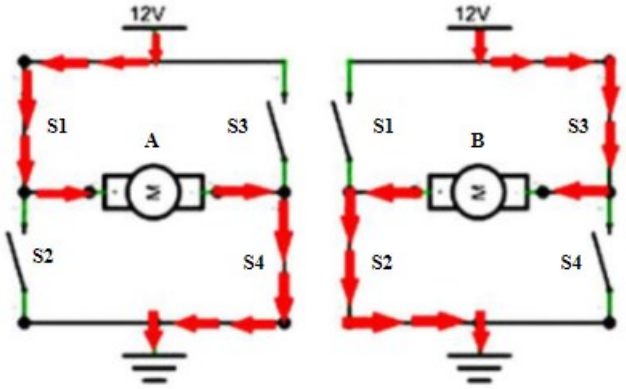
- Cầu H:

- Khóa S: 0 là ngắt, 1 là đóng
- A là quay theo Clockwise
- B là quay theo Counter Clockwise



S14	S23	A	B
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

2. Cầu H và Encoder



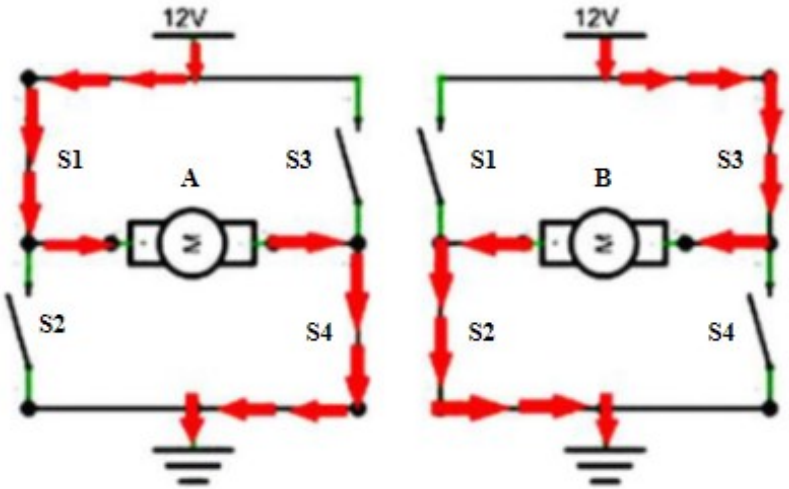
- Tín hiệu điều khiển: S14 và S23
- Tín hiệu đầu ra A và B

S14	S23	A	B
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

2. Cầu H và Encoder

Tín hiệu điều khiển: S14 và S23

- 2 PWM

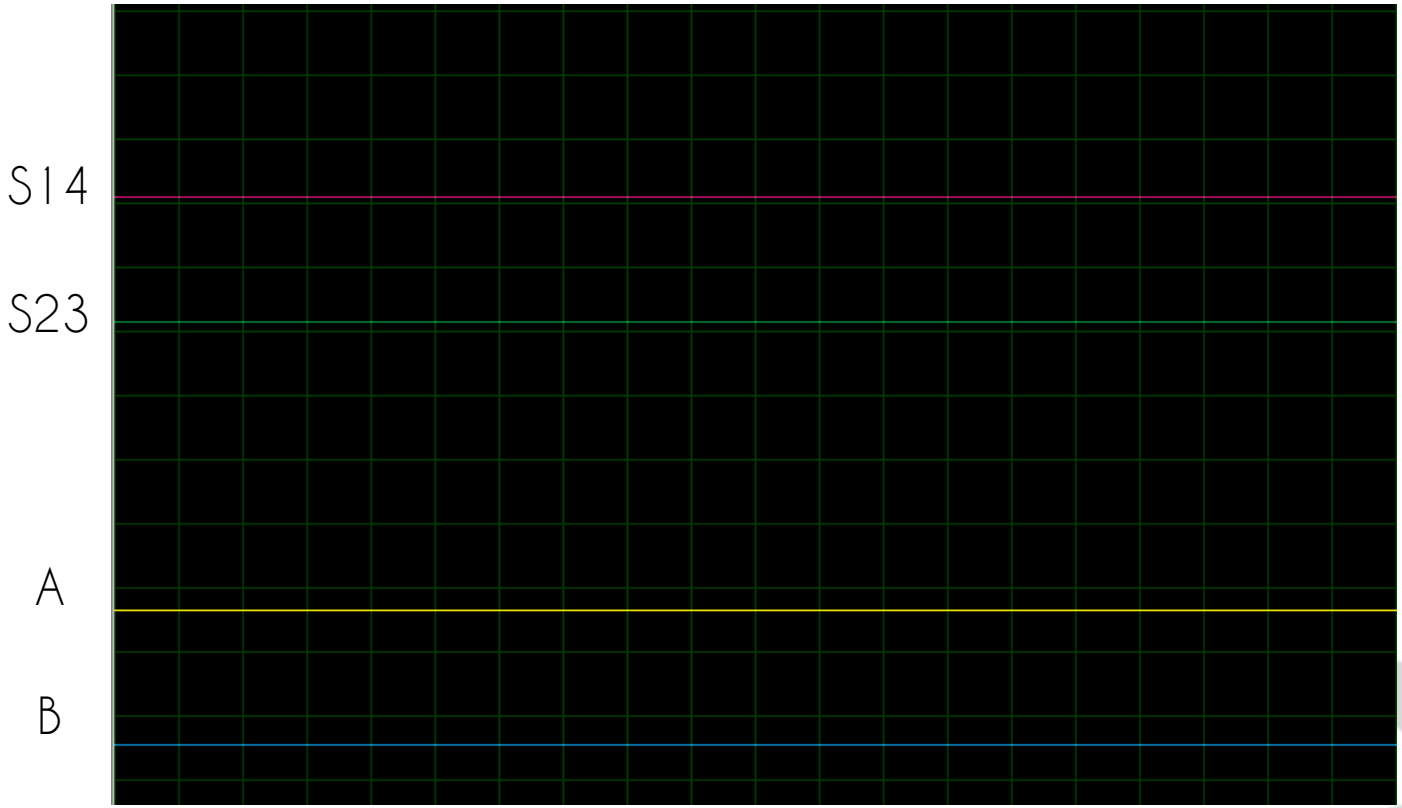


2. Cầu H và Encoder

Tín hiệu	Loại kết nối	Mục đích
S14	PWM	Chọn chiều và điều khiển tốc độ
S23	PWM	Chọn chiều và điều khiển tốc độ

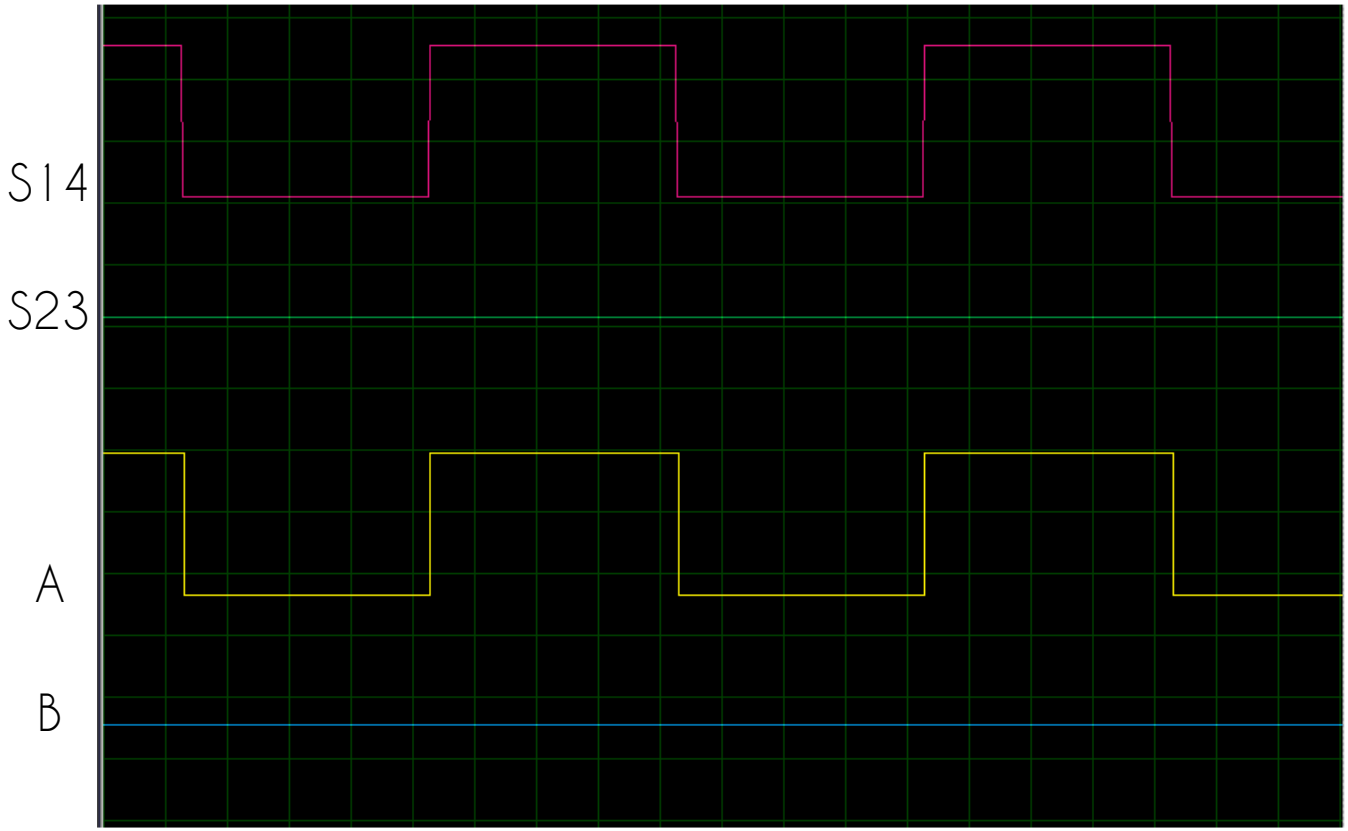
2. Cầu H và Encoder

S14 0% và S23 0%



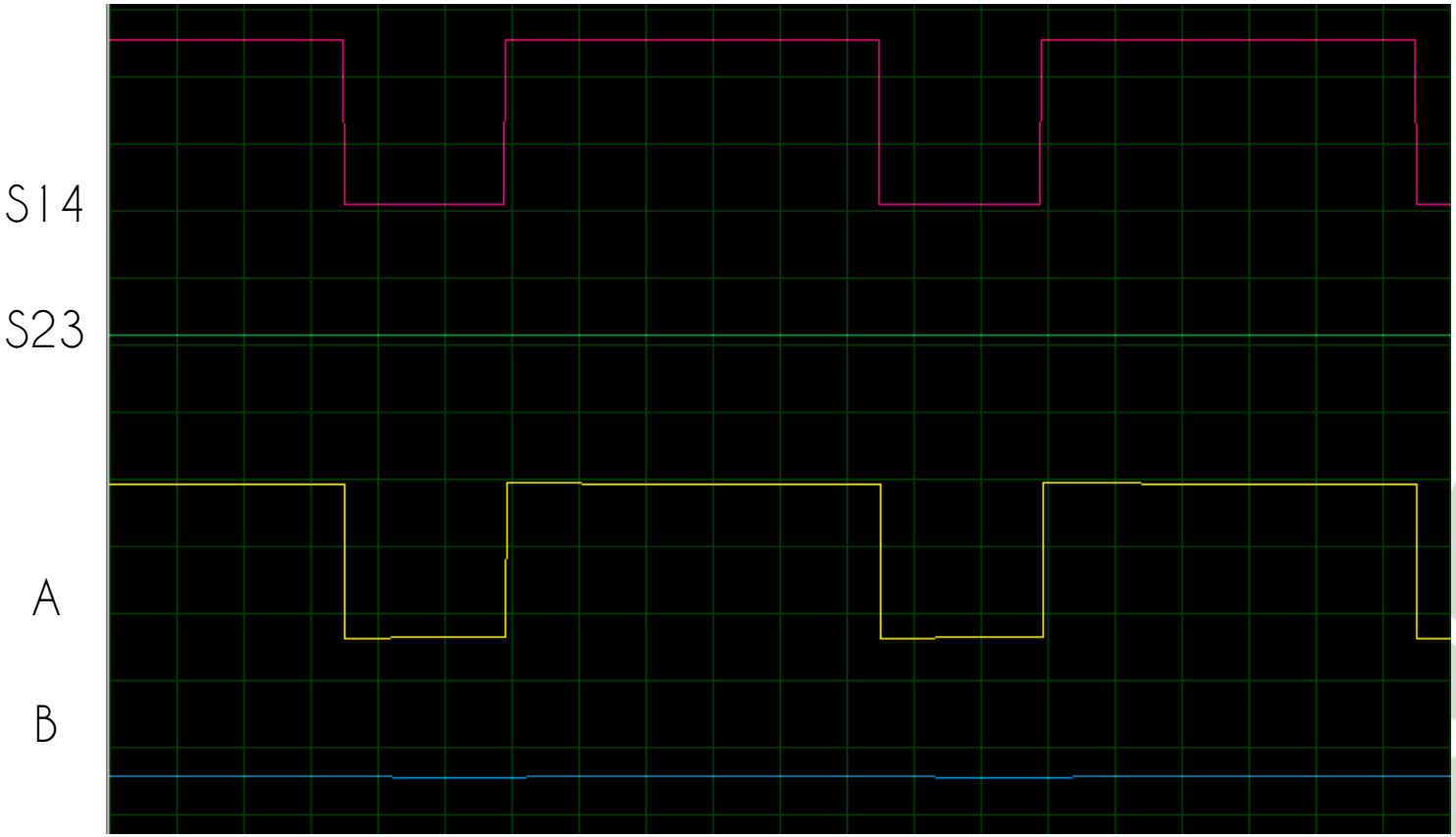
2. Cầu H và Encoder

S14 50% và S23 0%



2. Cầu H và Encoder

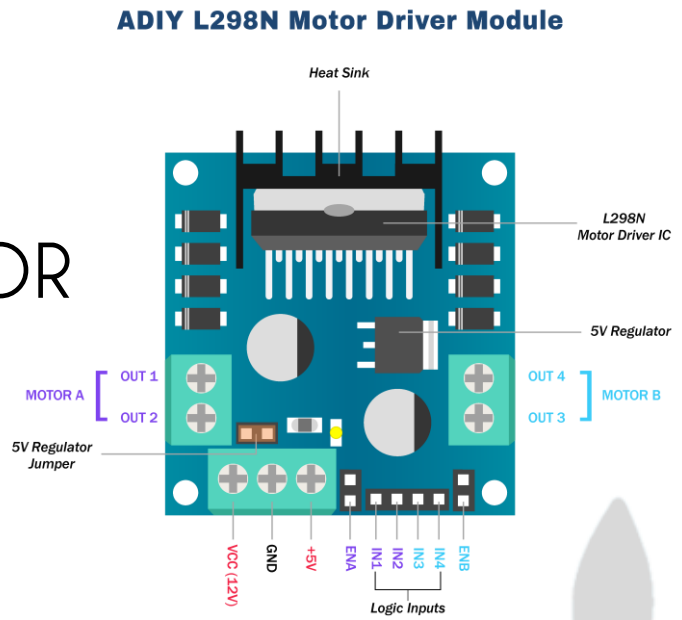
S14 70% và S23 0%



2. Cầu H và Encoder

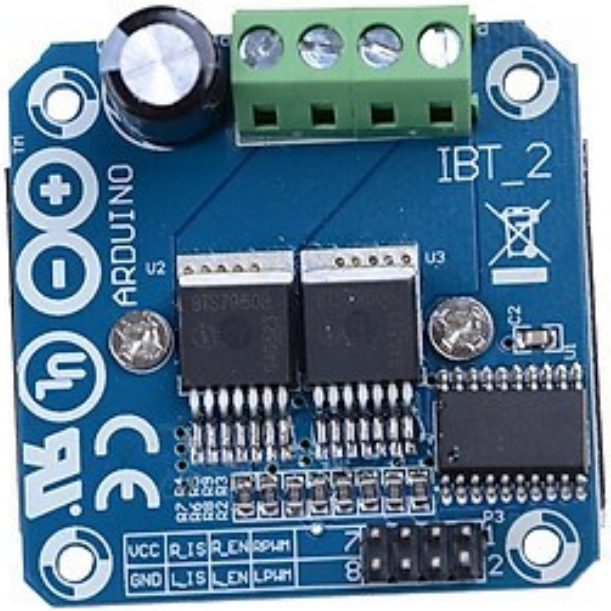
- Cầu H thông dụng: L298N

- IN1: S14
- IN2: S23
- OUT1 và OUT2: DC MOTOR
- Tương tự cho IN3, IN4 và OUT3, OUT4
- Công suất 25W



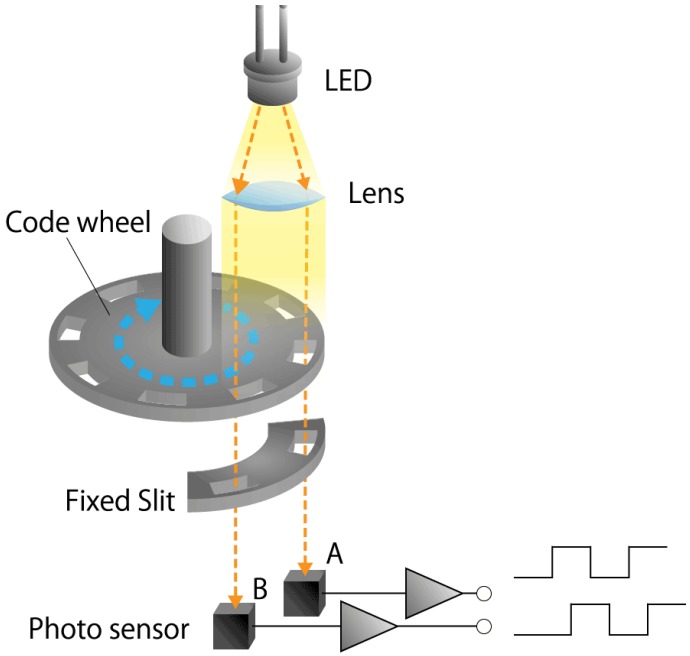
2. Cầu H và Encoder

- Cầu H thông dụng: BTS7960
- R_PWM, L_PWM: S14 , S23
- Công suất: 43A



2. Cầu H và Encoder

- Encoder

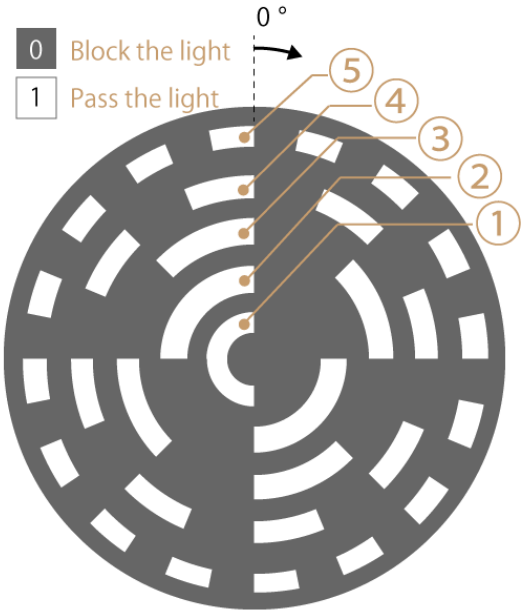


2. Cầu H và Encoder

- Incremental Encoder: Tăng giá trị khi xoay, cung cấp thông tin về sự thay đổi vị trí, sử dụng xung và bộ đếm.
- Absolute Encoder: Cung cấp thông tin vị trí chính xác, mỗi vị trí có giá trị tuyệt đối riêng biệt, giữ nguyên thông tin sau mất điện hoặc khởi động lại.

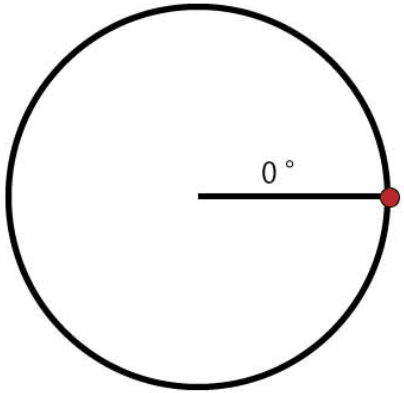
2. Cầu H và Encoder

- Absolute Encoder

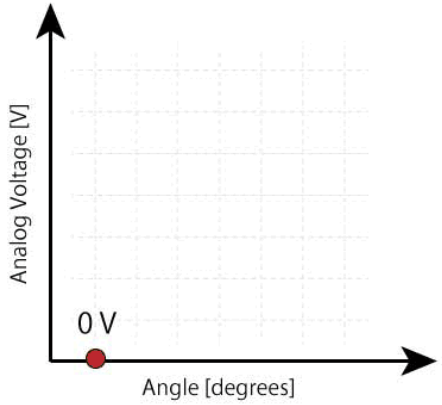


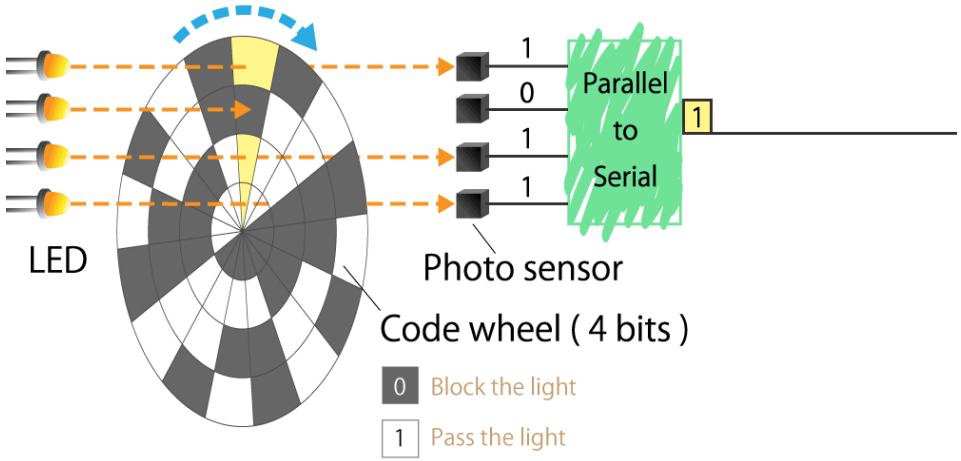
Angle [degrees]	①	②	③	④	⑤
0.00	0	0	0	0	0
11.25	0	0	0	0	1
22.50	0	0	0	1	0
33.75	0	0	0	1	1
45.00	0	0	1	0	0
56.25	0	0	1	0	1
67.50	0	0	1	1	0
78.75	0	0	1	1	1
90.00	0	1	0	0	0
·	·	·	·	·	·
·	·	·	·	·	·
·	·	·	·	·	·
315.00	1	1	1	0	0
326.25	1	1	1	0	1
337.50	1	1	1	1	0
348.75	1	1	1	1	1

2. Cầu H và Encoder



Angle [degrees]	Digital Binary Code ① ② ③	Analog Voltage [V]
0	0 0 0	0.00
45	0 0 1	0.63
90	0 1 0	1.25
135	0 1 1	1.88
180	1 0 0	2.50
225	1 0 1	3.13
270	1 1 0	3.75
315	1 1 1	4.38

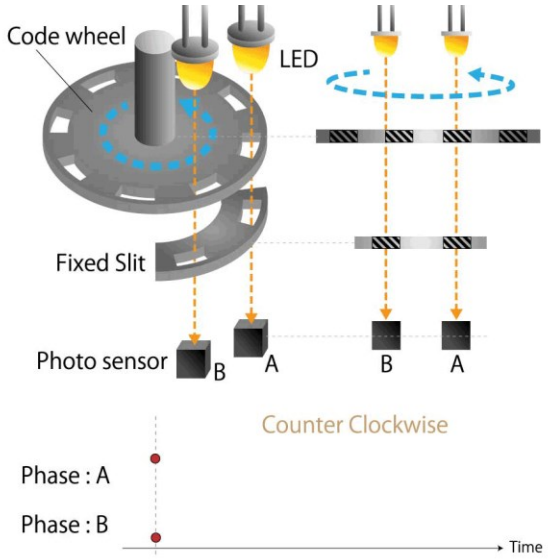




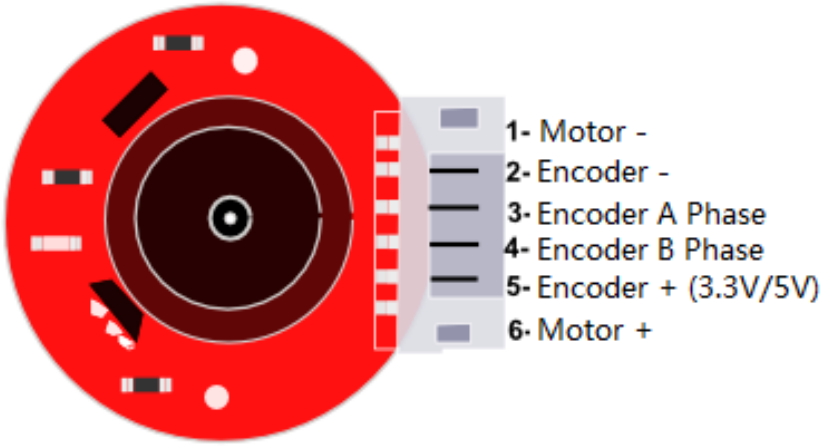
2. Cầu H và Encoder

- Incremental Encoder:

- Dựa vào tín hiệu xung trả về ta có thể xác định được chiều quay, tốc độ và vị trí của trục động cơ.

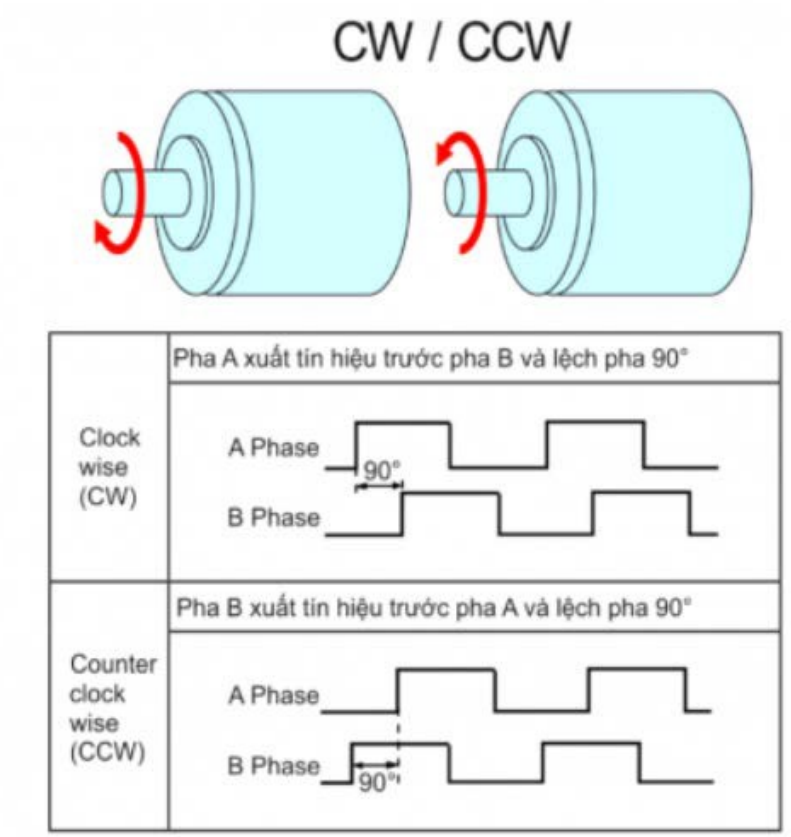


2. Cầu H và Encoder



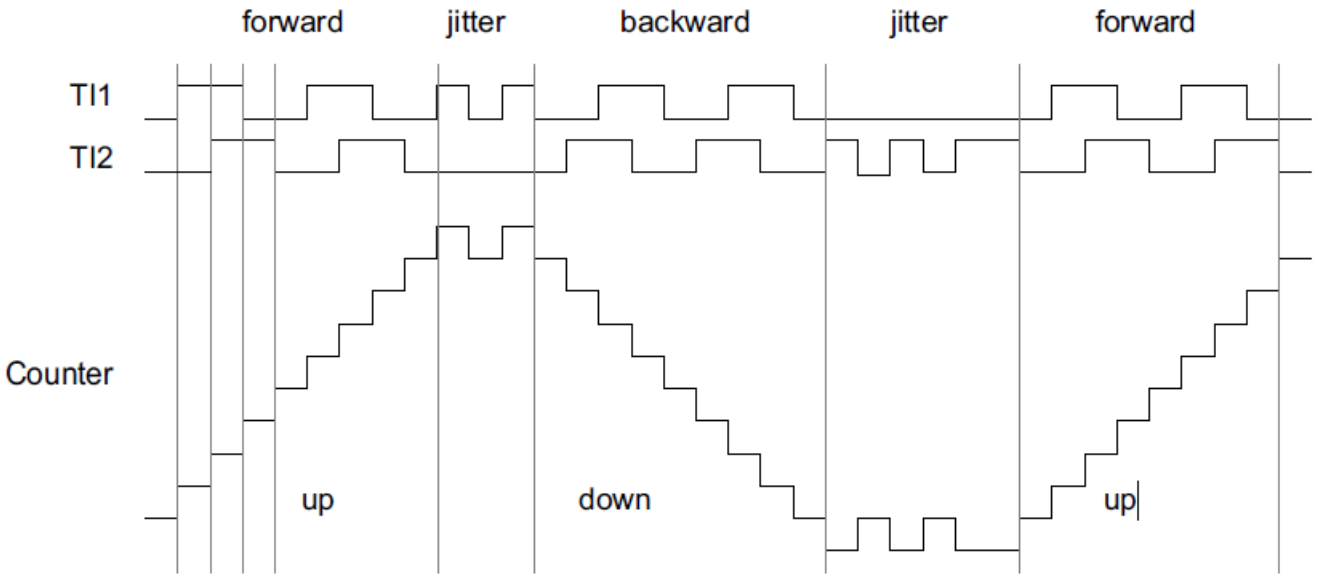
2. Cầu H và Encoder

- Chiều quay: Khi encoder quay theo chiều kim đồng hồ (CW), xung A sẽ xuất tín hiệu trước xung B và lệch pha 90 độ. Ngược lại, khi encoder quay ngược chiều kim đồng hồ (CCW), xung B sẽ xuất tín hiệu trước và lệch pha 90 độ so với xung A.



2. Cầu H và Encoder

- Tốc độ và vị trí: dựa vào số xung trả về để xác định.



2. Cầu H và Encoder

- PPR(Pulse per round/Resolution) là số xung A hoặc B trên 1 vòng quay
- Cách đọc encoder:

Vận tốc = Số xung trả về trong 1 khoảng thời gian

Vị trí = Tích phân vận tốc

2. Cầu H và Encoder

- Ví dụ:

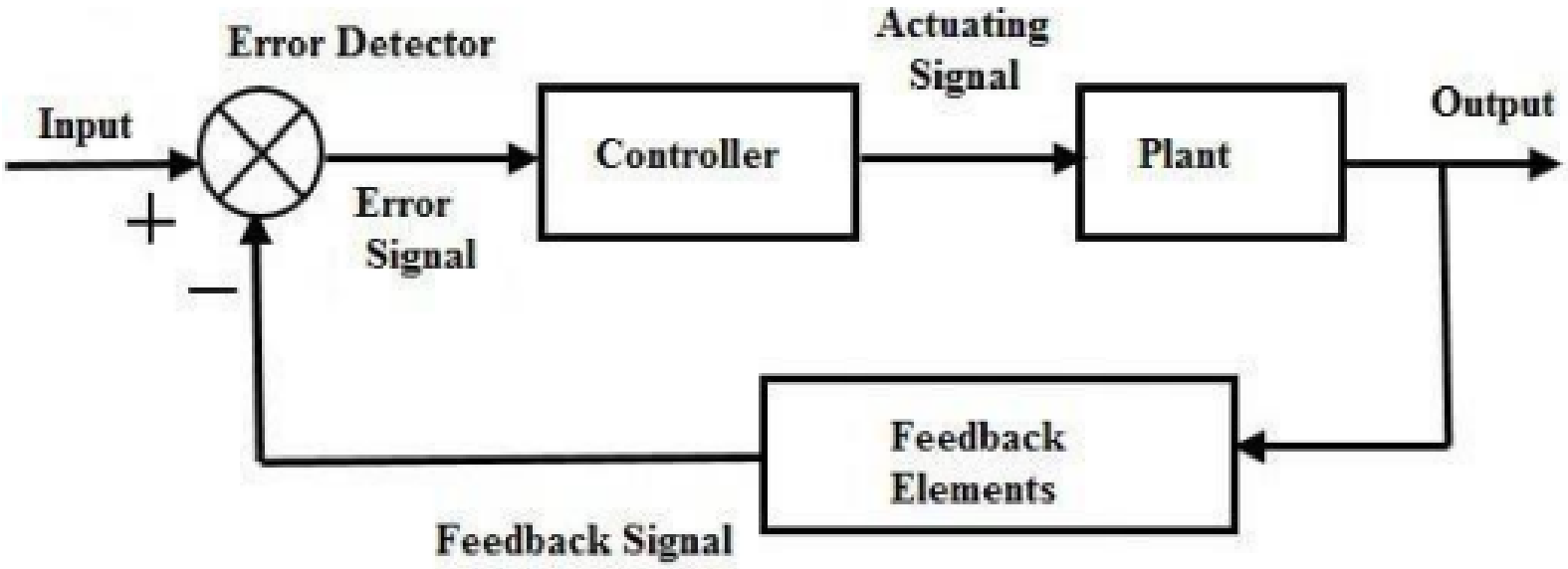
- PPR: 1000
- Ngắt 10 ms
- Đọc được 20 xung
- Vận tốc: 2 xung/ms

$$2/1000 \times 1000 \times 360 = 720(\deg ree/s)$$

- Vị trí: Cộng thêm: $2/1000 \times 360 = 0.72(\deg ree)$

3. Bộ điều khiển vòng kín và bộ điều khiển PID

- Bộ điều khiển vòng kín:

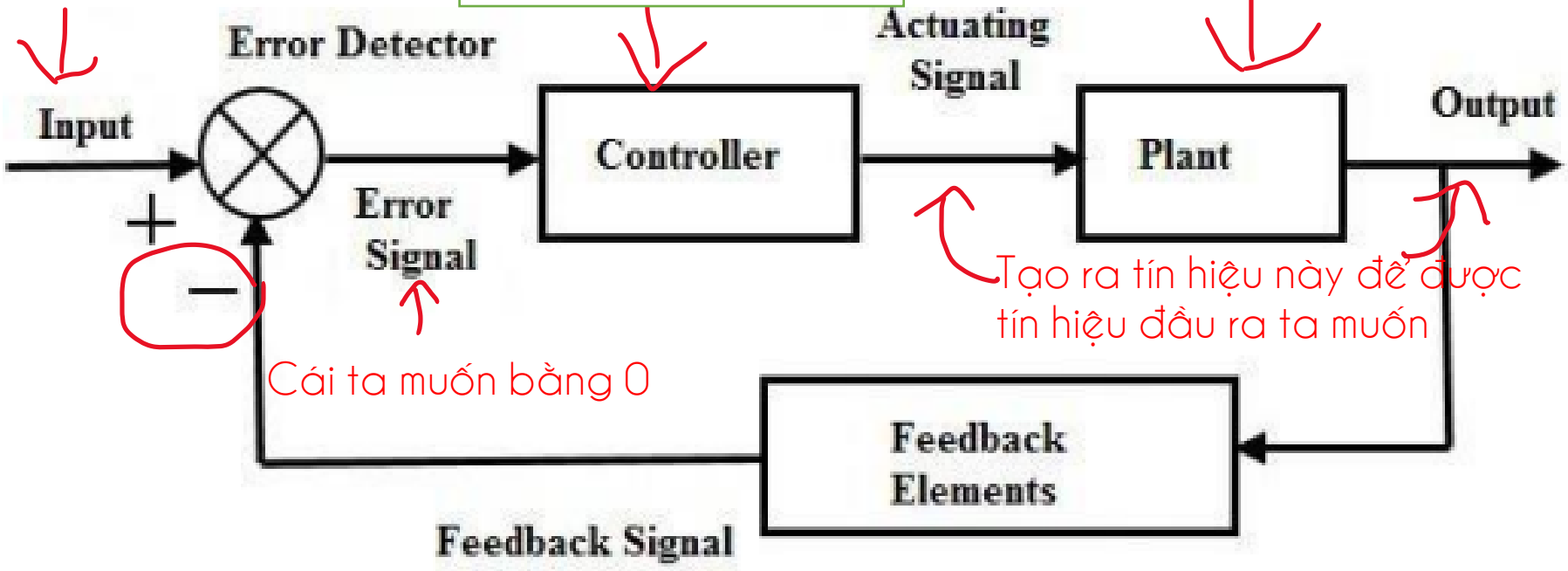


3. Bộ điều khiển vòng kín và bộ điều khiển PID

Cái mà ta muốn hệ thống thực hiện

Đẩy sai số điều khiển về 0 *theo thời gian*

Hệ thống ta muốn điều khiển



Tạo ra tín hiệu này để được tín hiệu đầu ra ta muốn

Cái ta muốn bằng 0

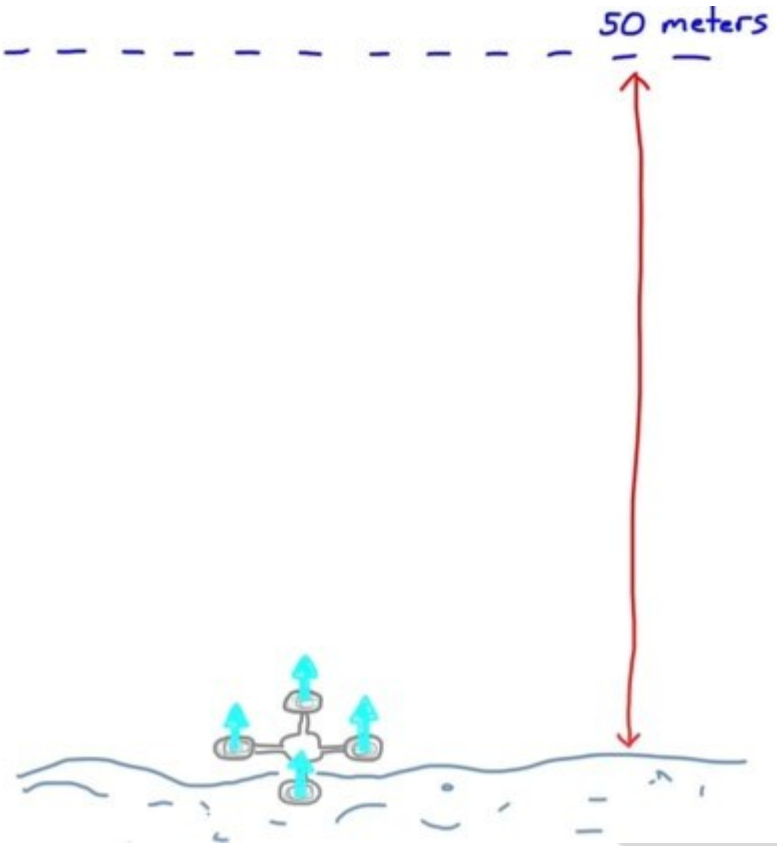
3. Bộ điều khiển vòng kín và bộ điều khiển PID

- Sai số điều khiển (error signal)

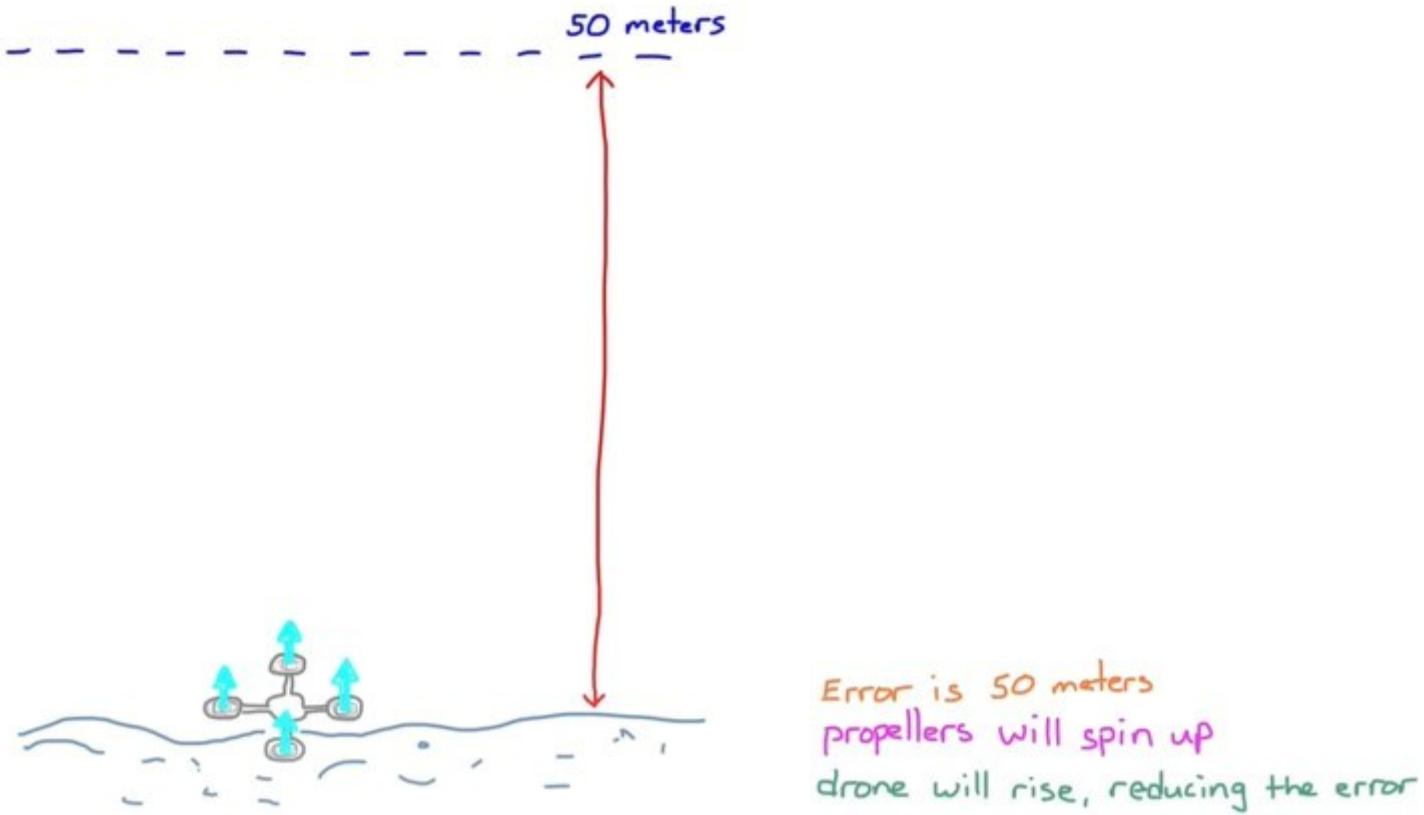
Error value = Output value – Input value

3. Bộ điều khiển vòng kín và bộ điều khiển PID

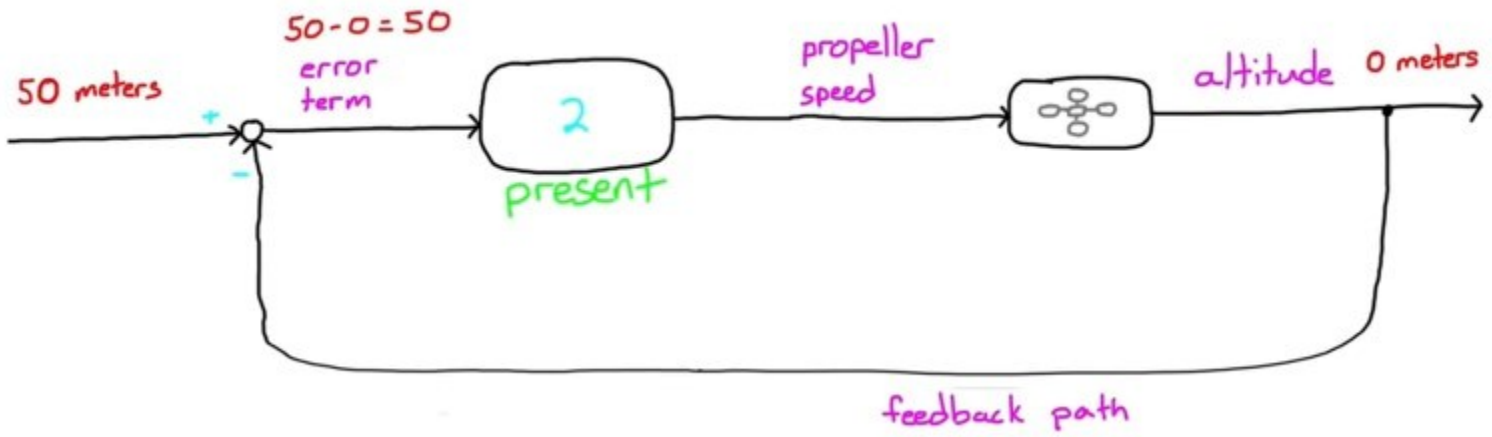
- Ví dụ: Giả sử 1 drone đang nằm trên mặt đất, và ta muốn nó bay lên độ cao 50m



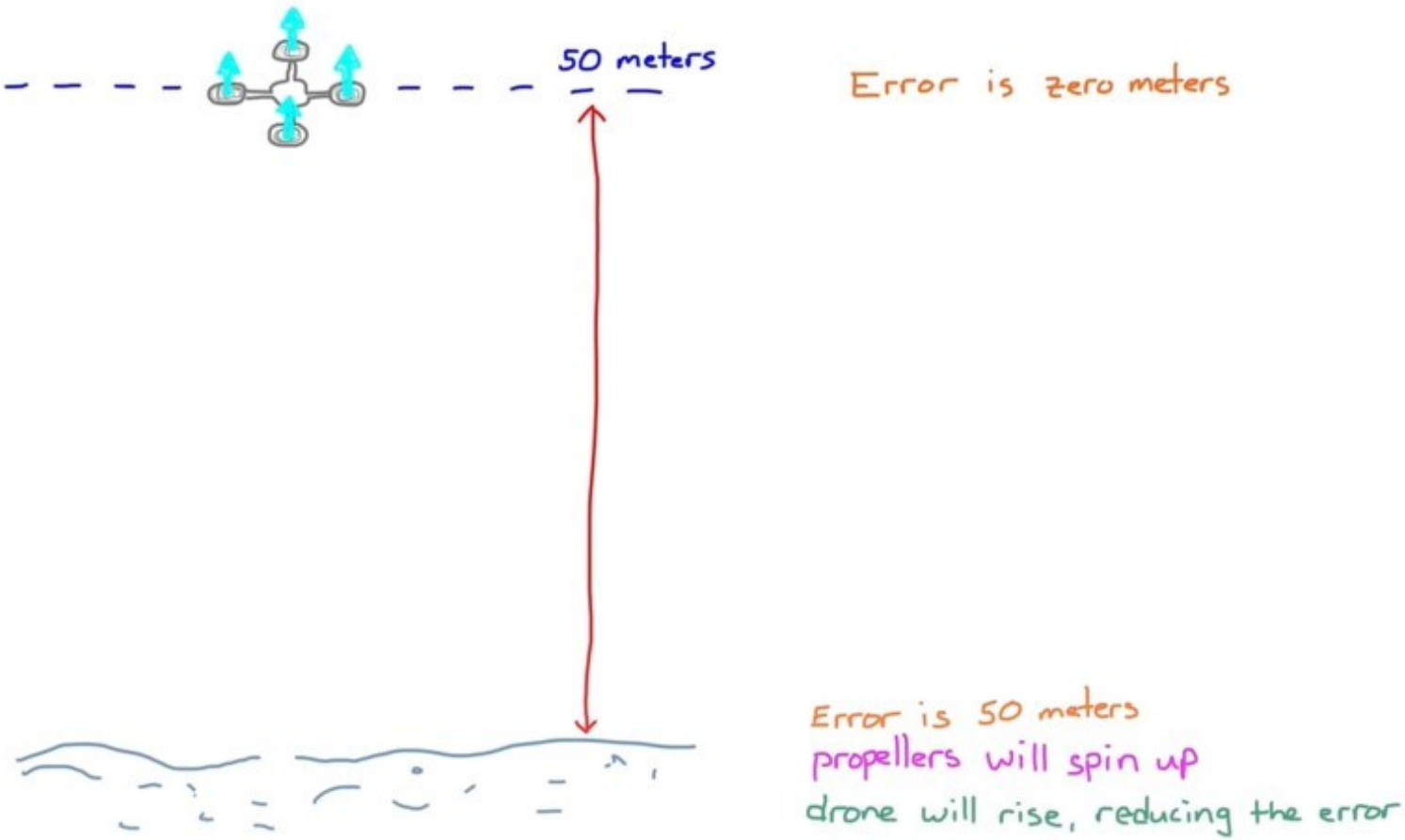
3. Bộ điều khiển vòng kín và bộ điều khiển PID



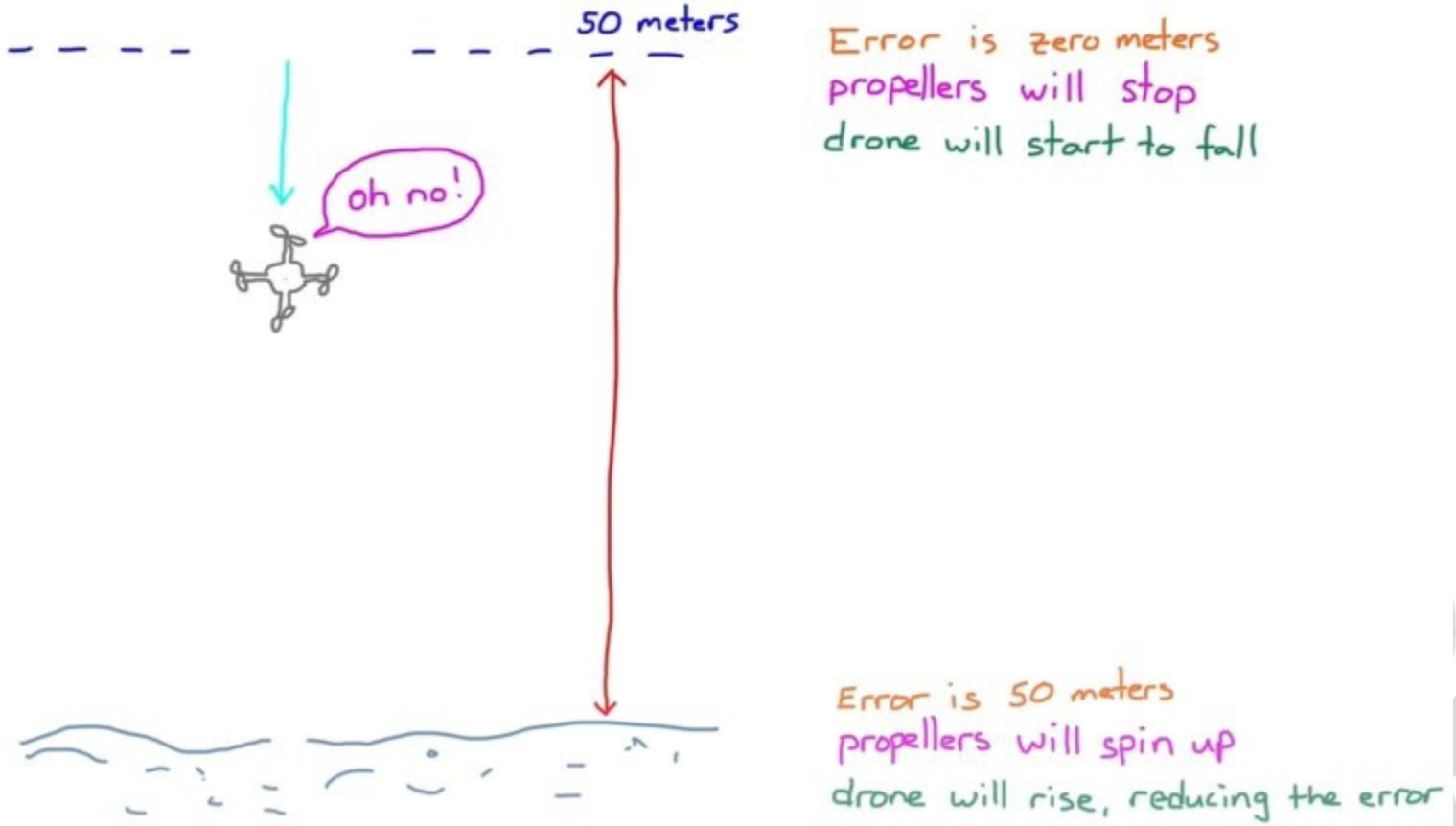
3. Bộ điều khiển vòng kín và bộ điều khiển PID



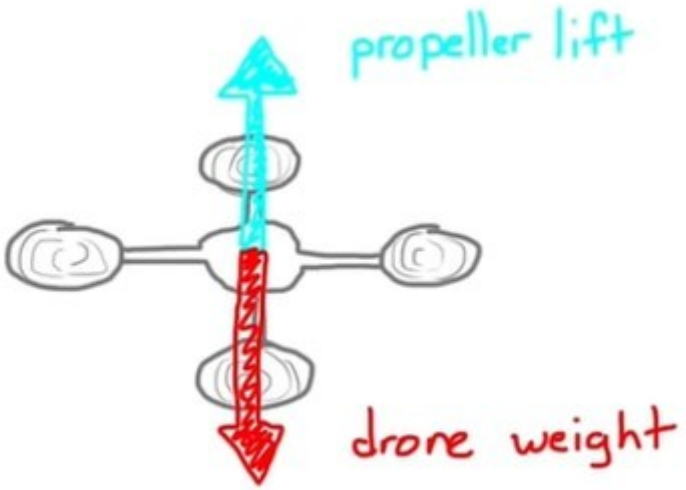
3. Bộ điều khiển vòng kín và bộ điều khiển PID



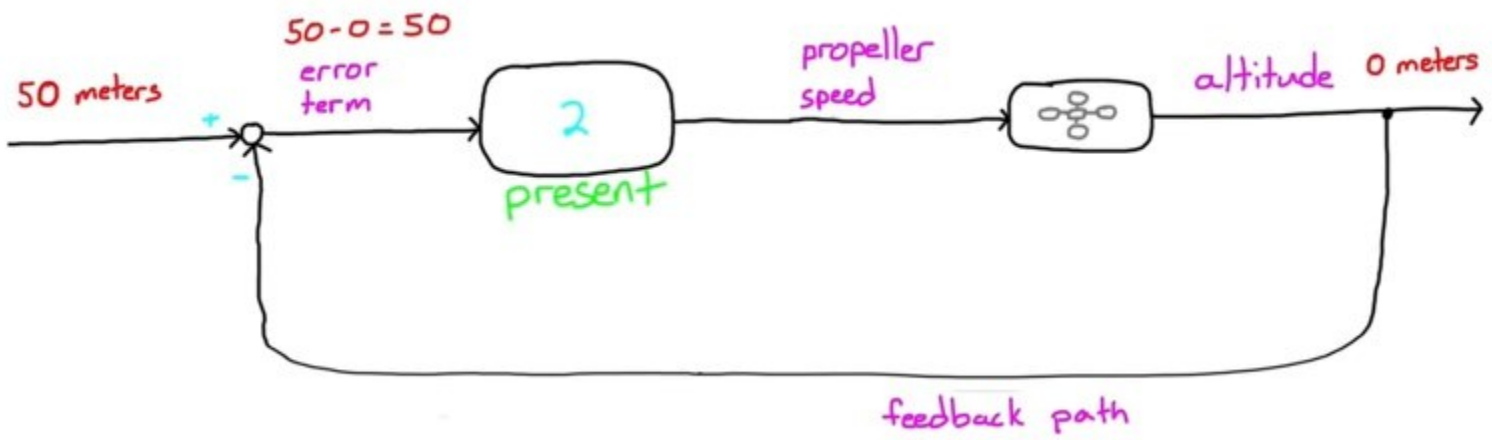
3. Bộ điều khiển vòng kín và bộ điều khiển PID



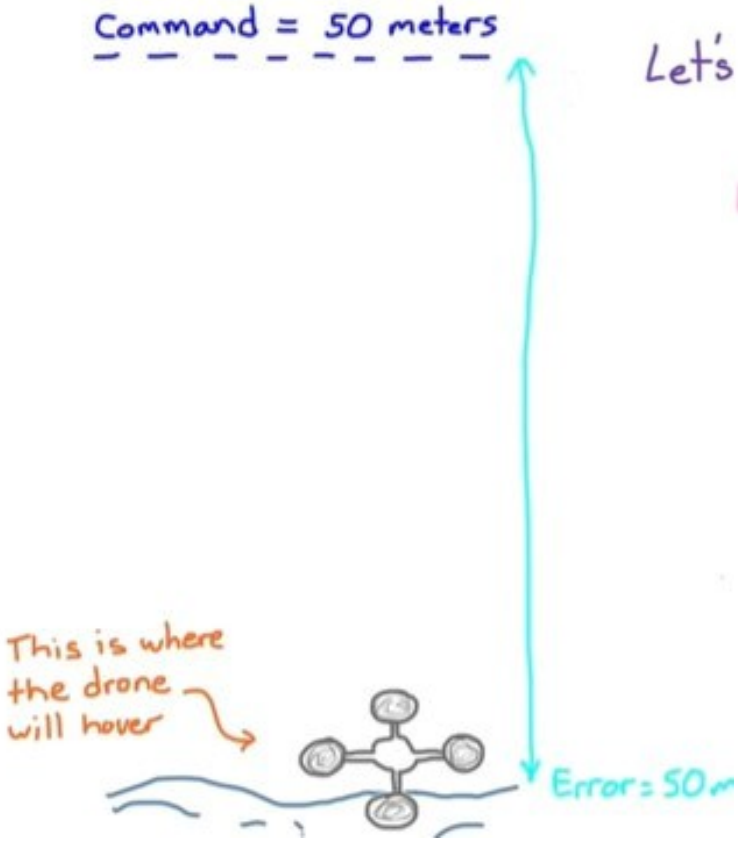
3. Bộ điều khiển vòng kín và bộ điều khiển PID



3. Bộ điều khiển vòng kín và bộ điều khiển PID



3. Bộ điều khiển vòng kín và bộ điều khiển PID

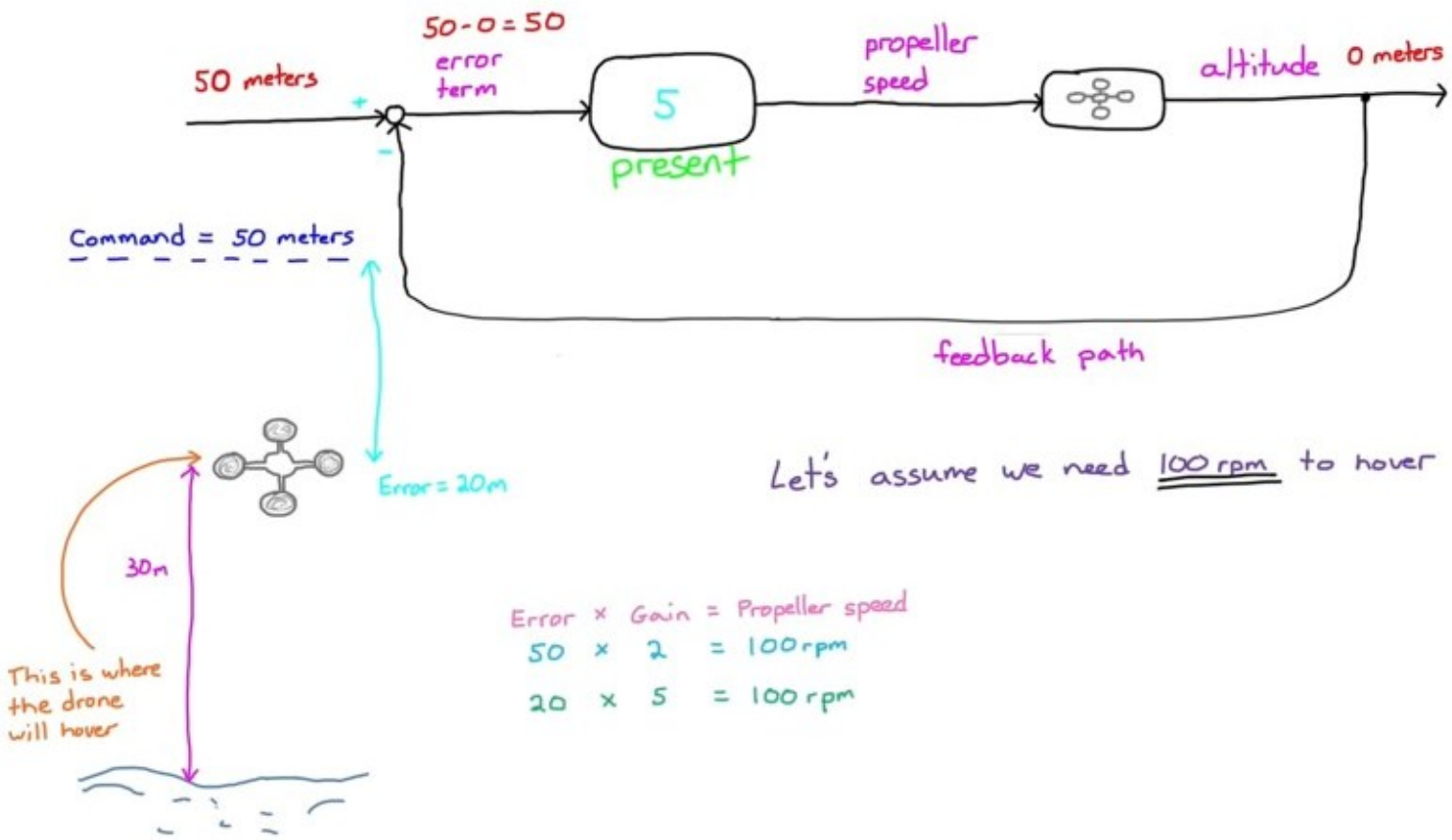


Let's assume we need 100 rpm to hover

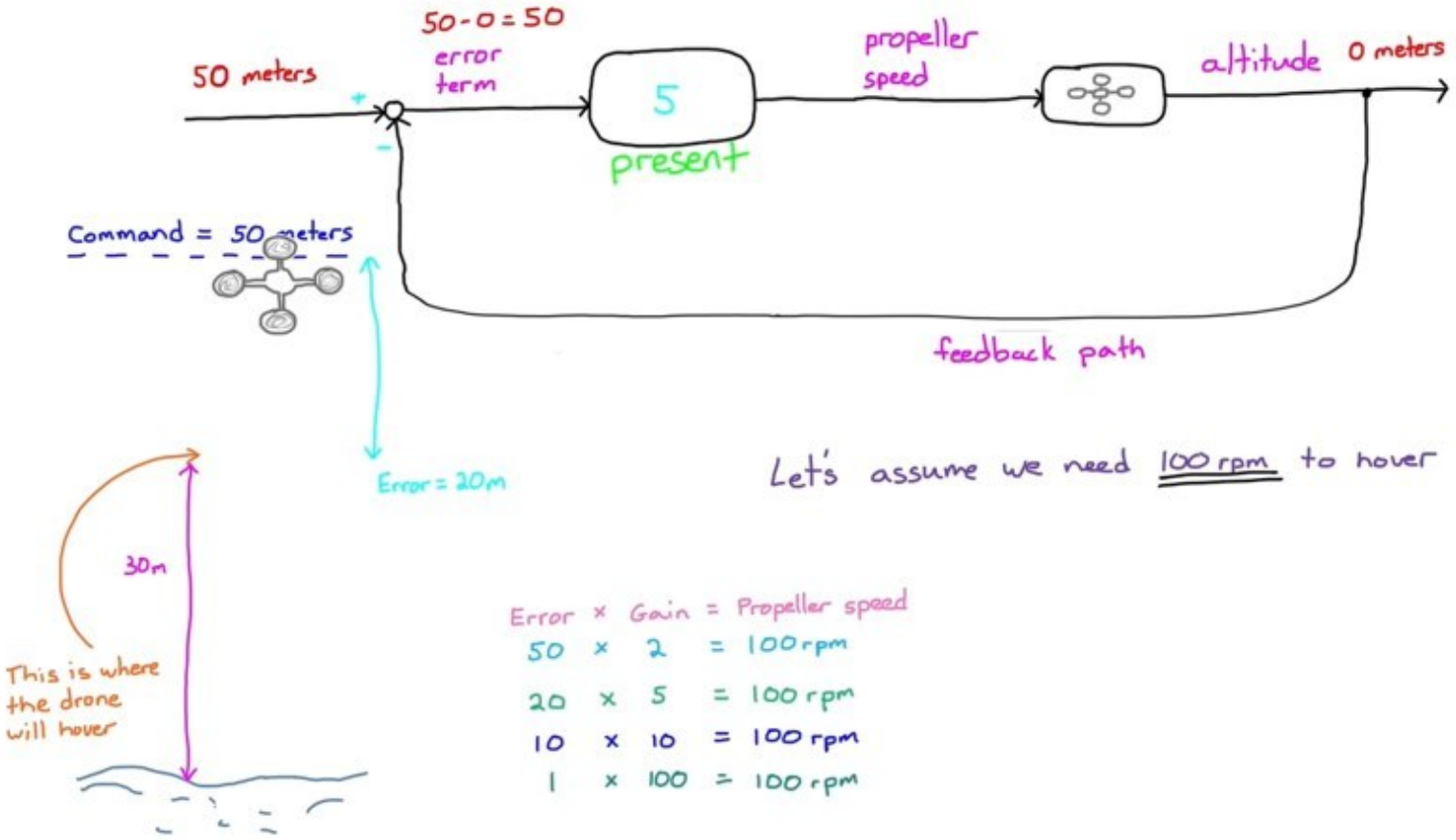
Error × Gain = Propeller speed

50 × 2 = 100 rpm

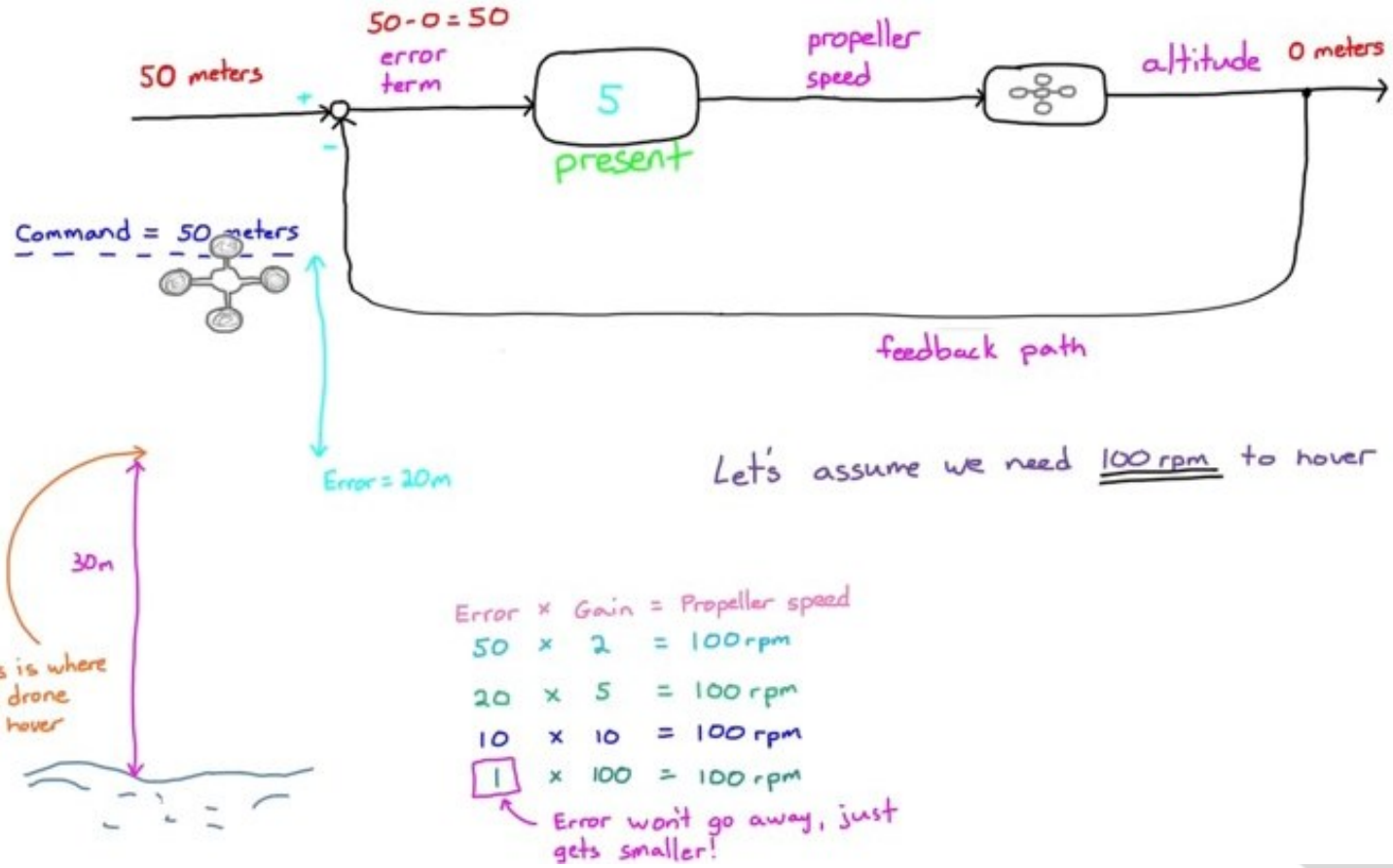
3. Bộ điều khiển vòng kín và bộ điều khiển PID



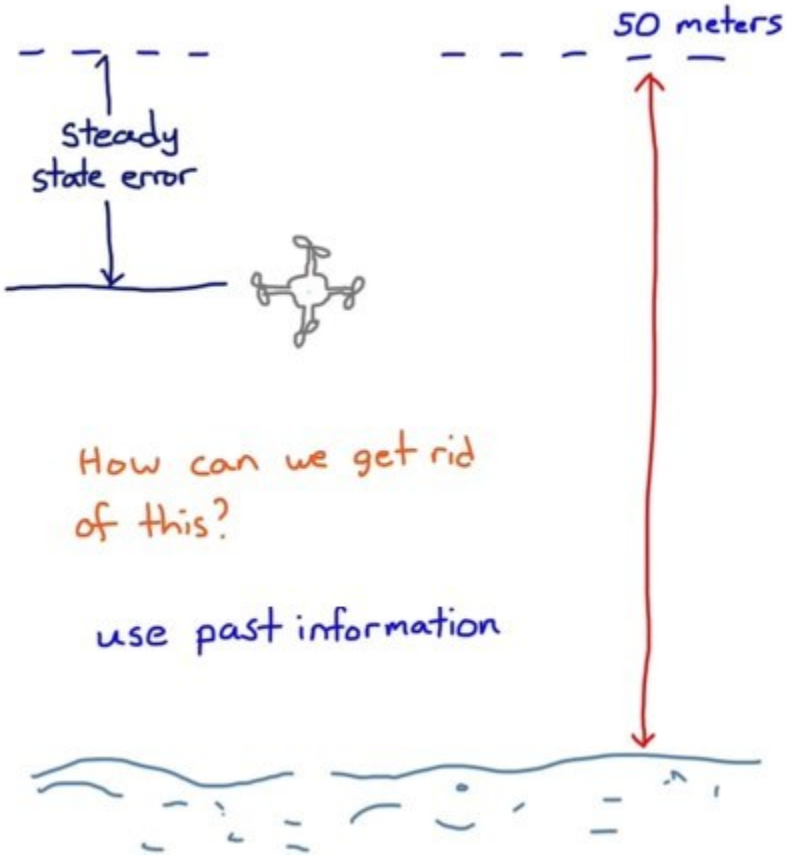
3. Bộ điều khiển vòng kín và bộ điều khiển PID



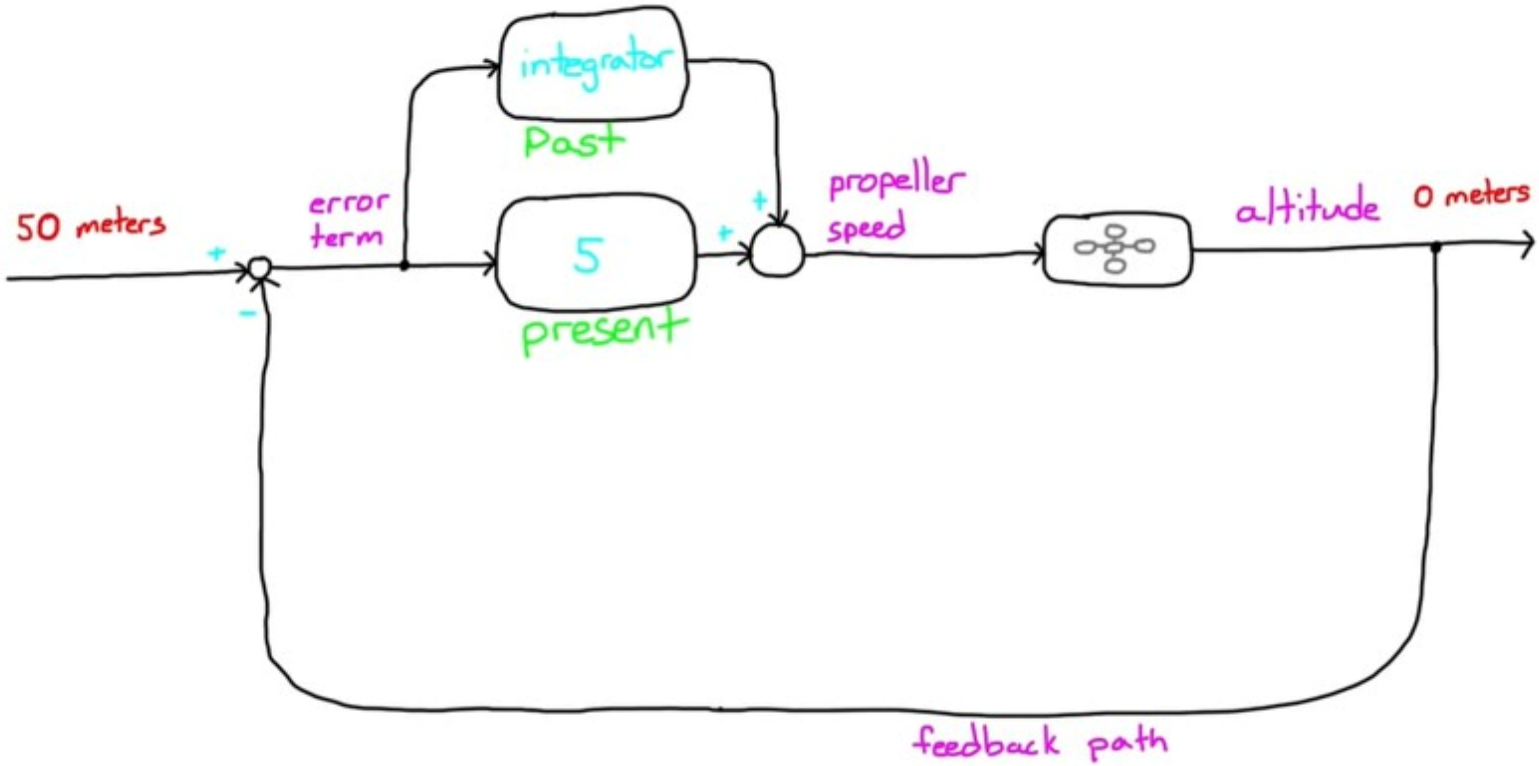
3. Bộ điều khiển vòng kín và bộ điều khiển PID



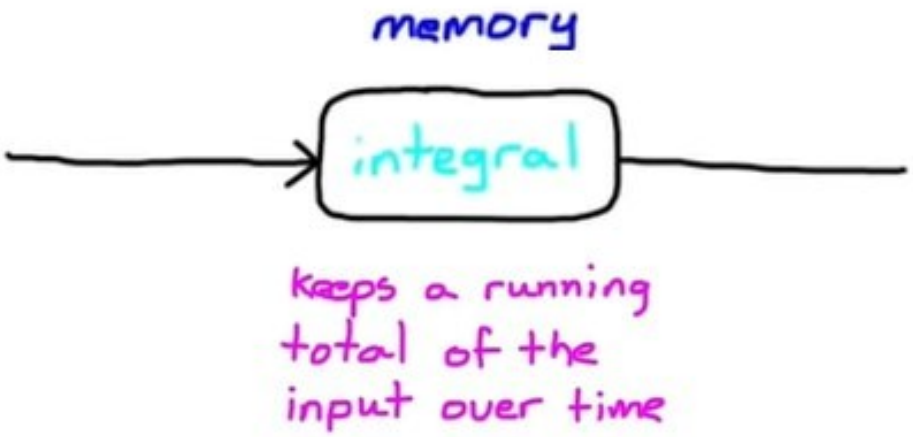
3. Bộ điều khiển vòng kín và bộ điều khiển PID



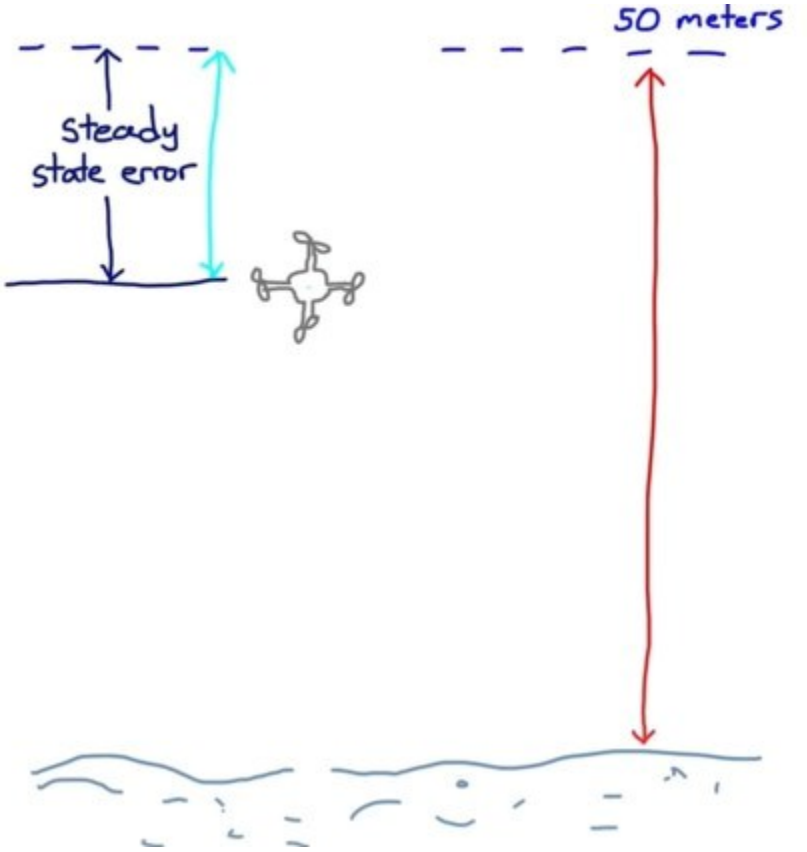
3. Bộ điều khiển vòng kín và bộ điều khiển PID

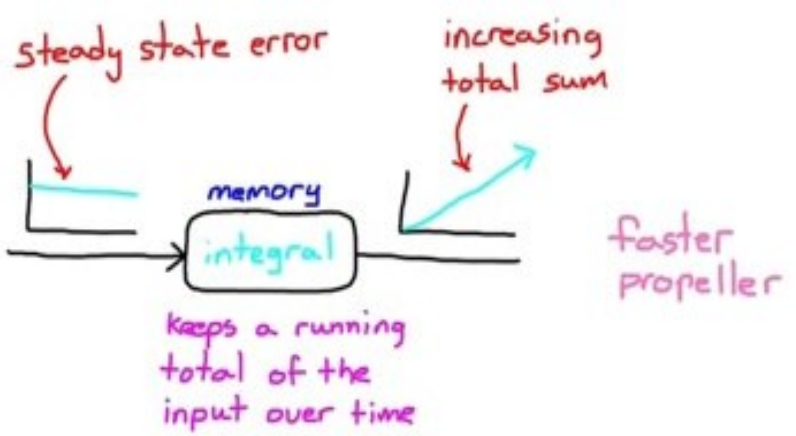


3. Bộ điều khiển vòng kín và bộ điều khiển PID

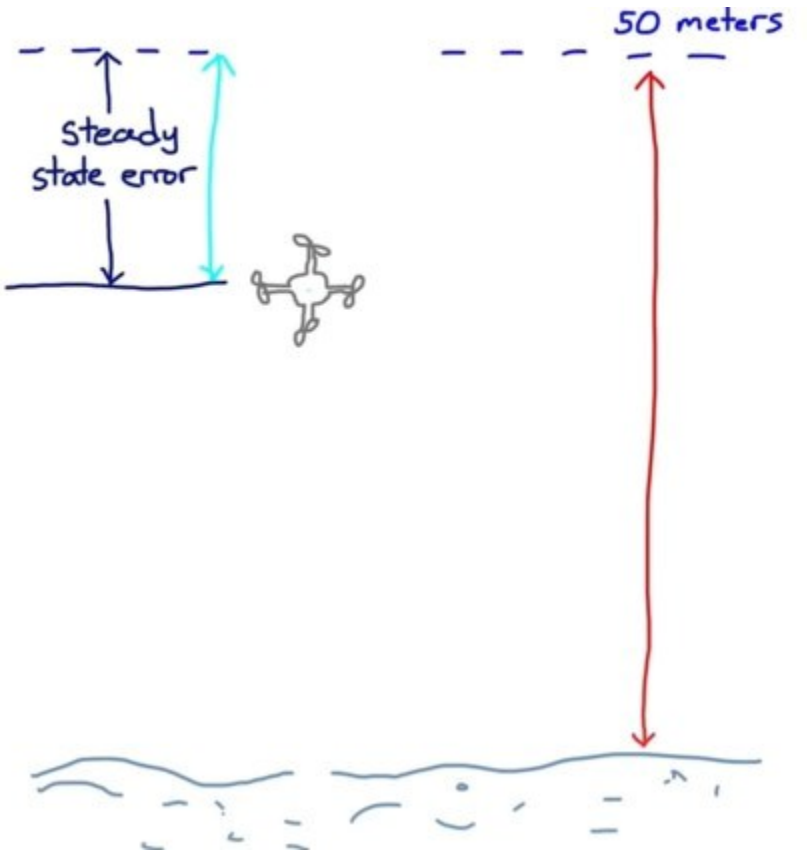


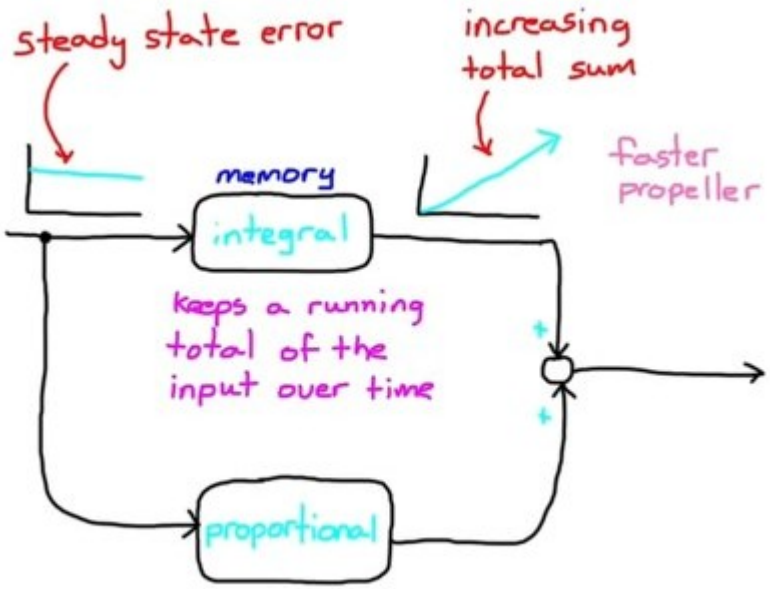
3. Bộ điều khiển vòng kín và bộ điều khiển PID



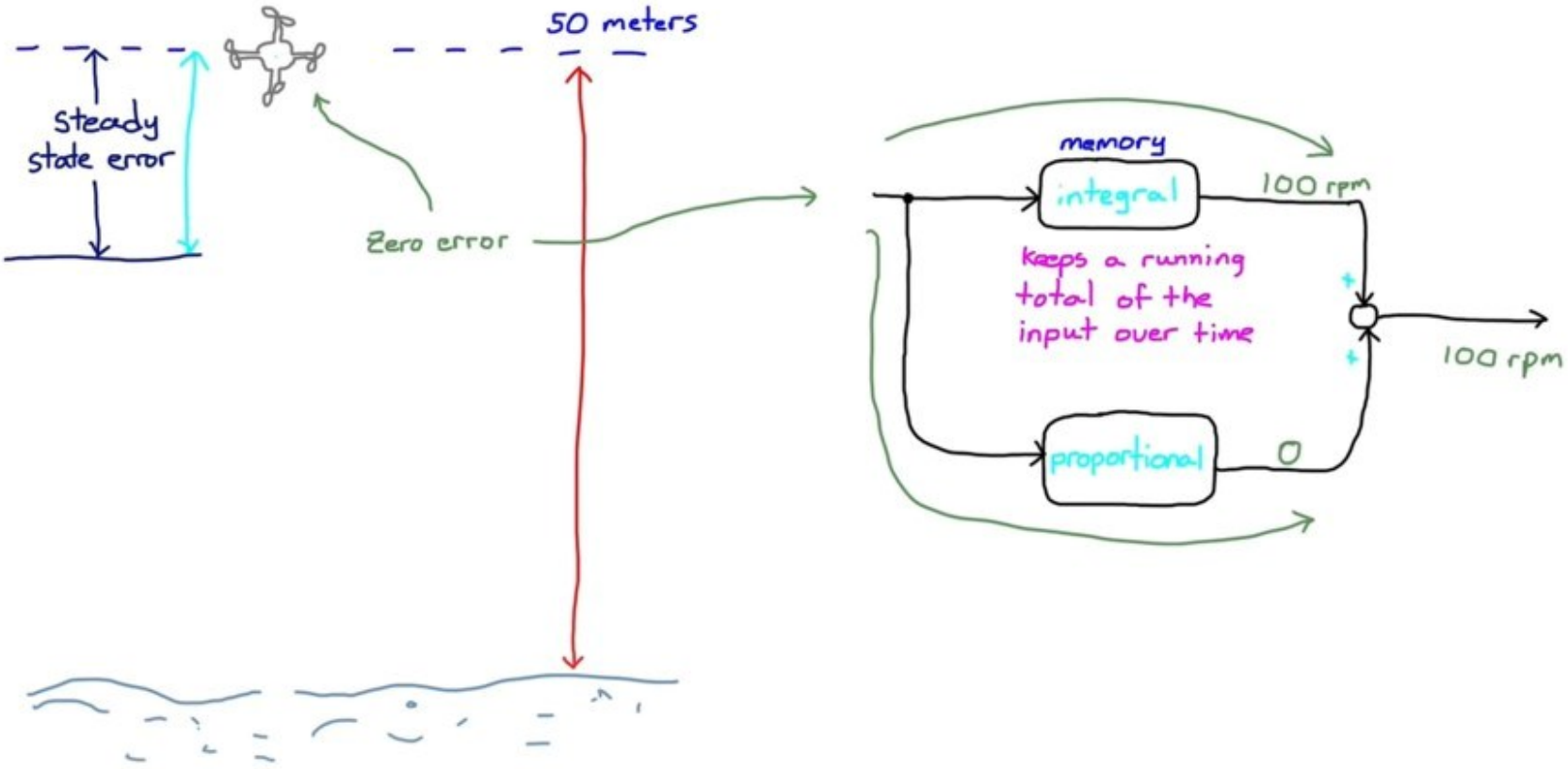


3. Bộ điều khiển vòng kín và bộ điều khiển PID

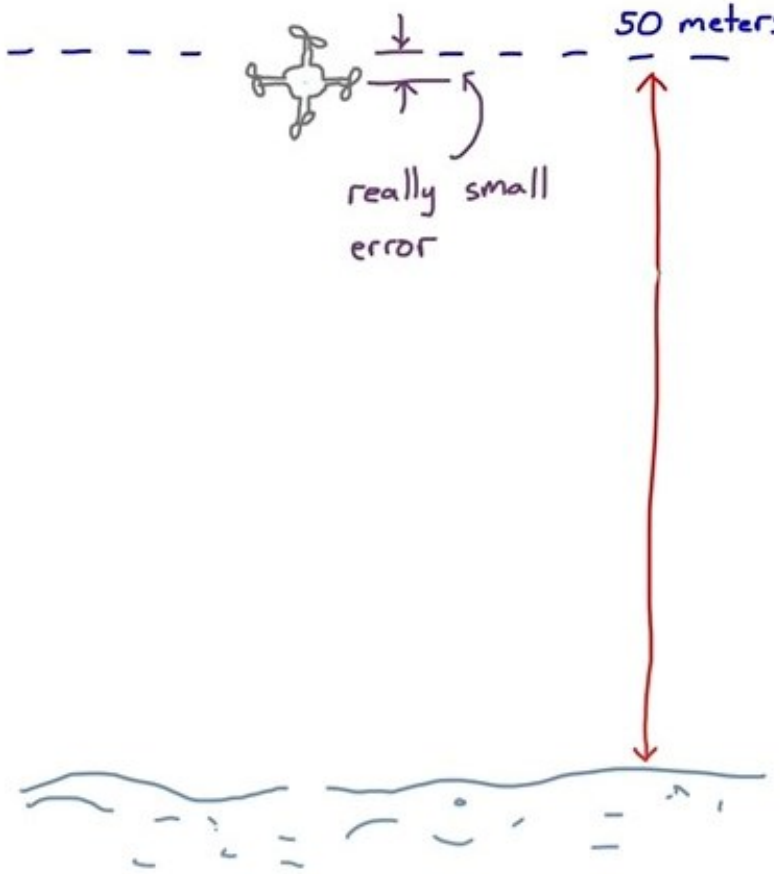


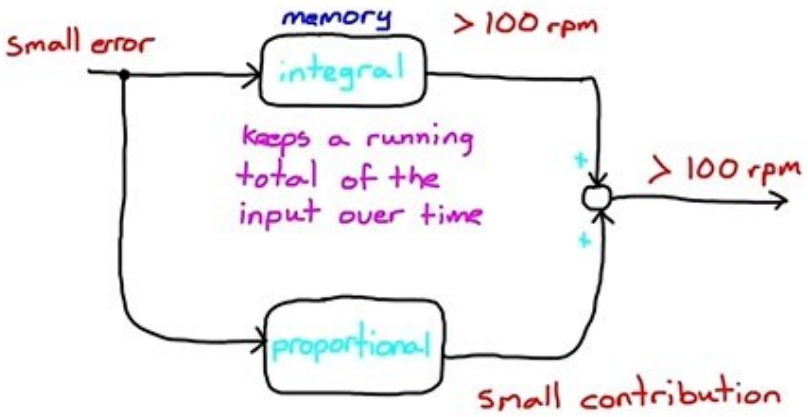


3. Bộ điều khiển vòng kín và bộ điều khiển PID

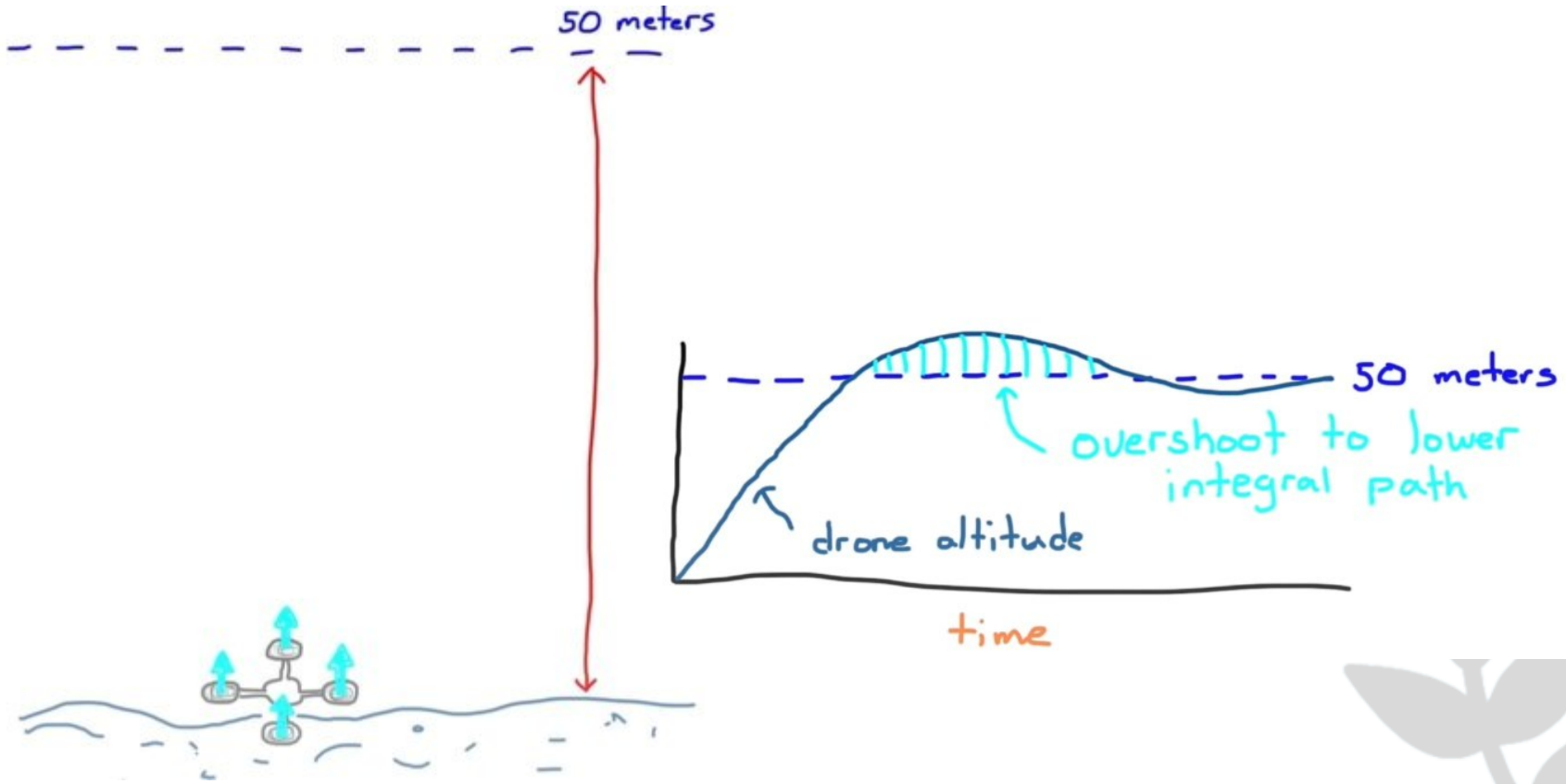


3. Bộ điều khiển vòng kín và bộ điều khiển PID

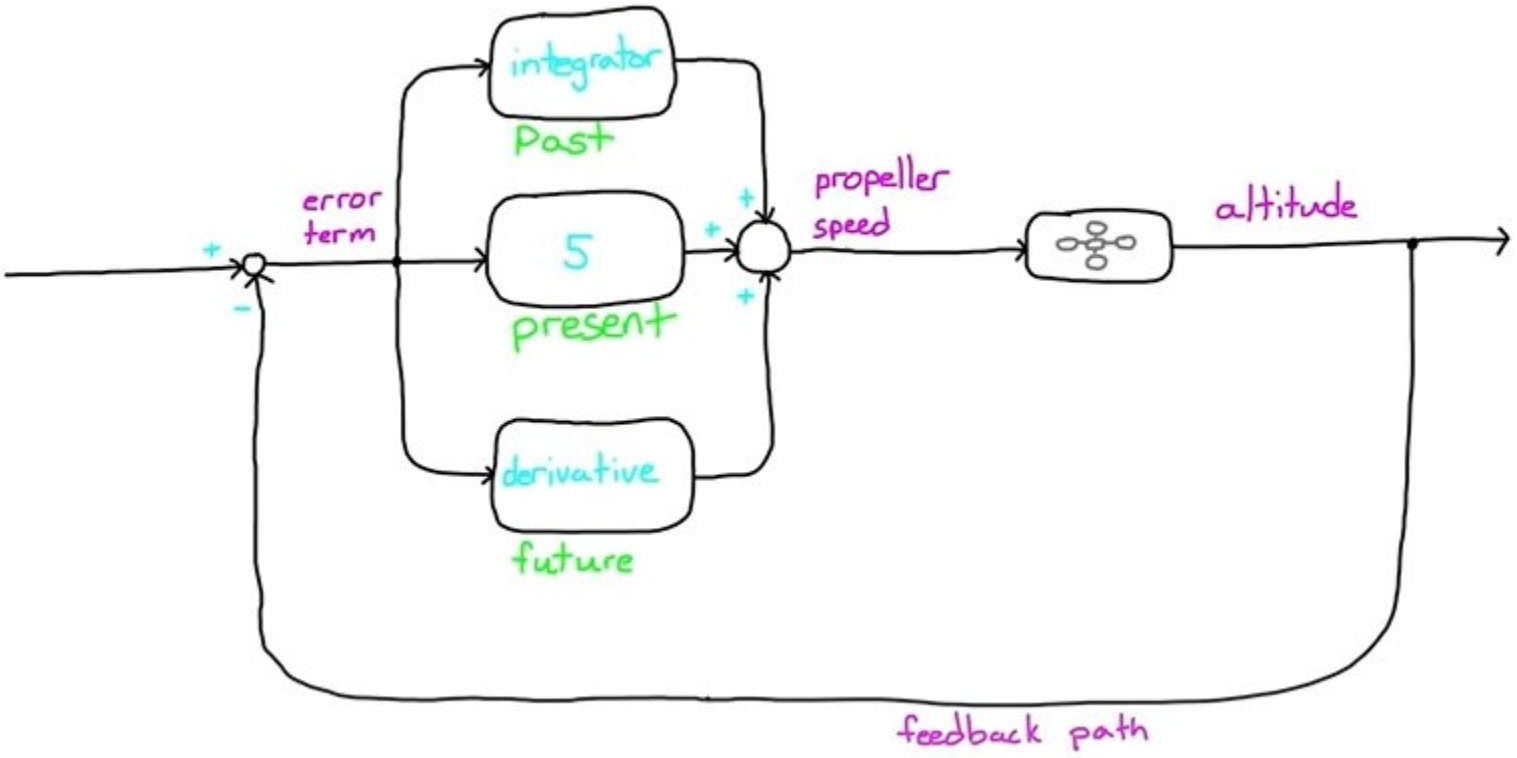




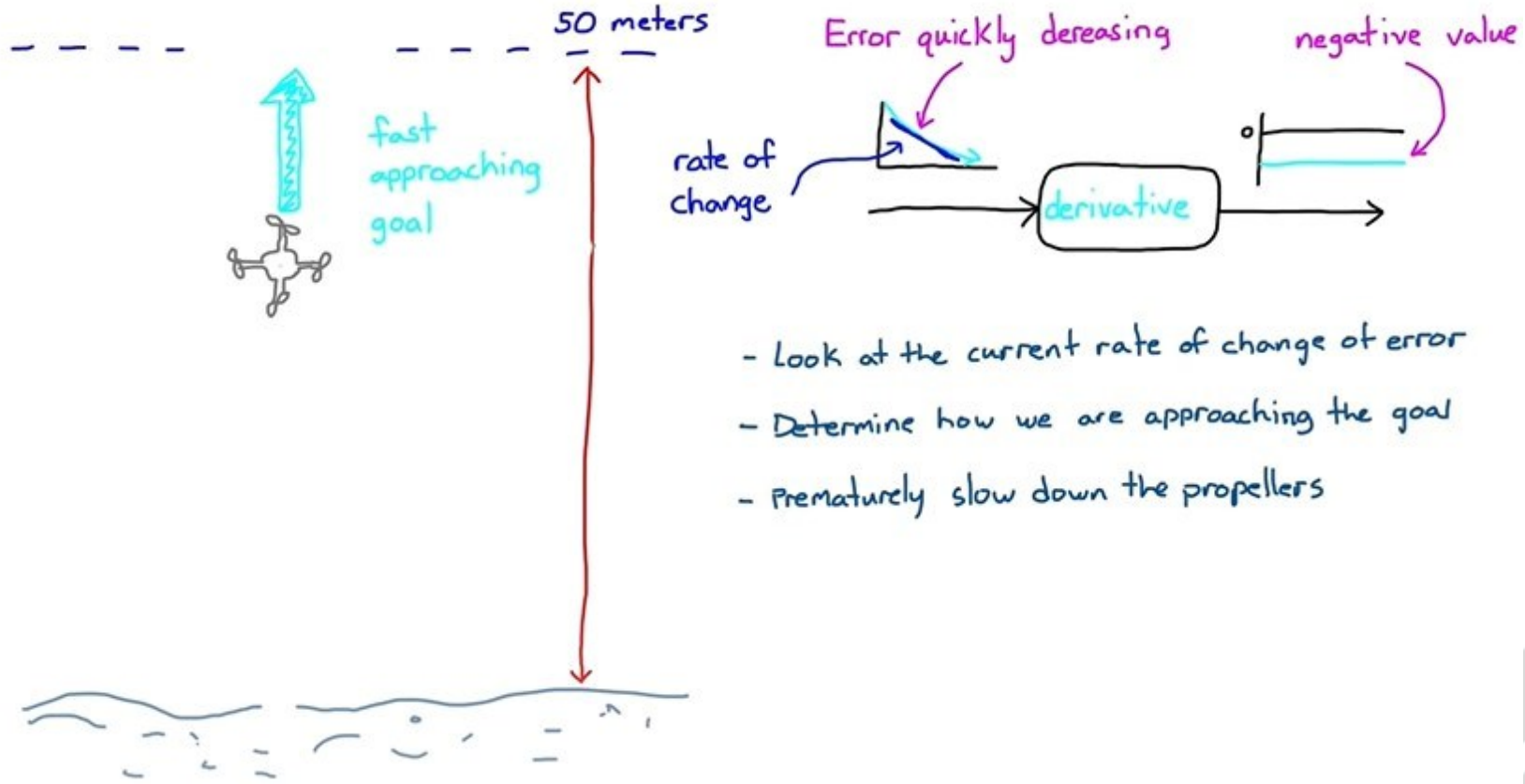
3. Bộ điều khiển vòng kín và bộ điều khiển PID



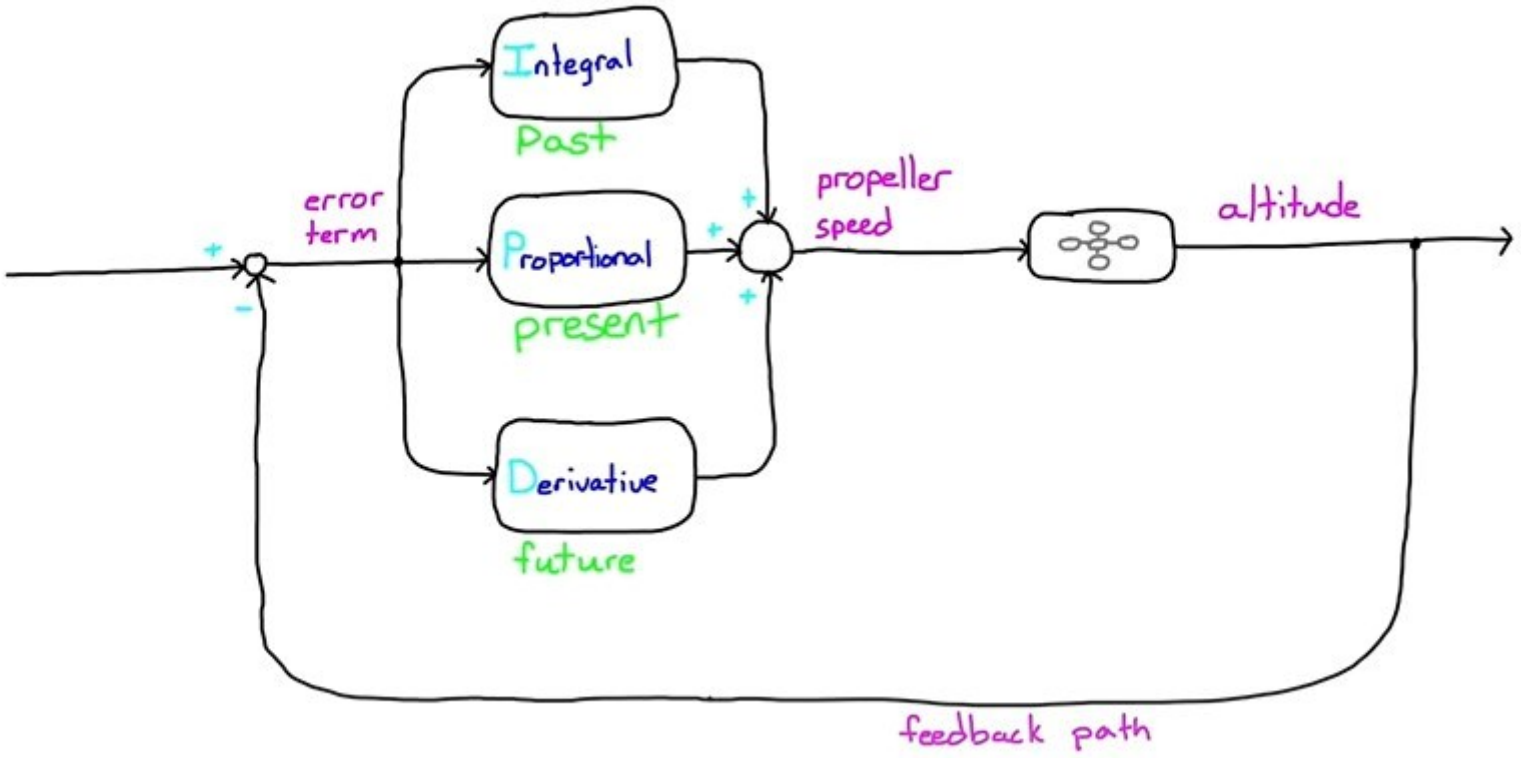
3. Bộ điều khiển vòng kín và bộ điều khiển PID



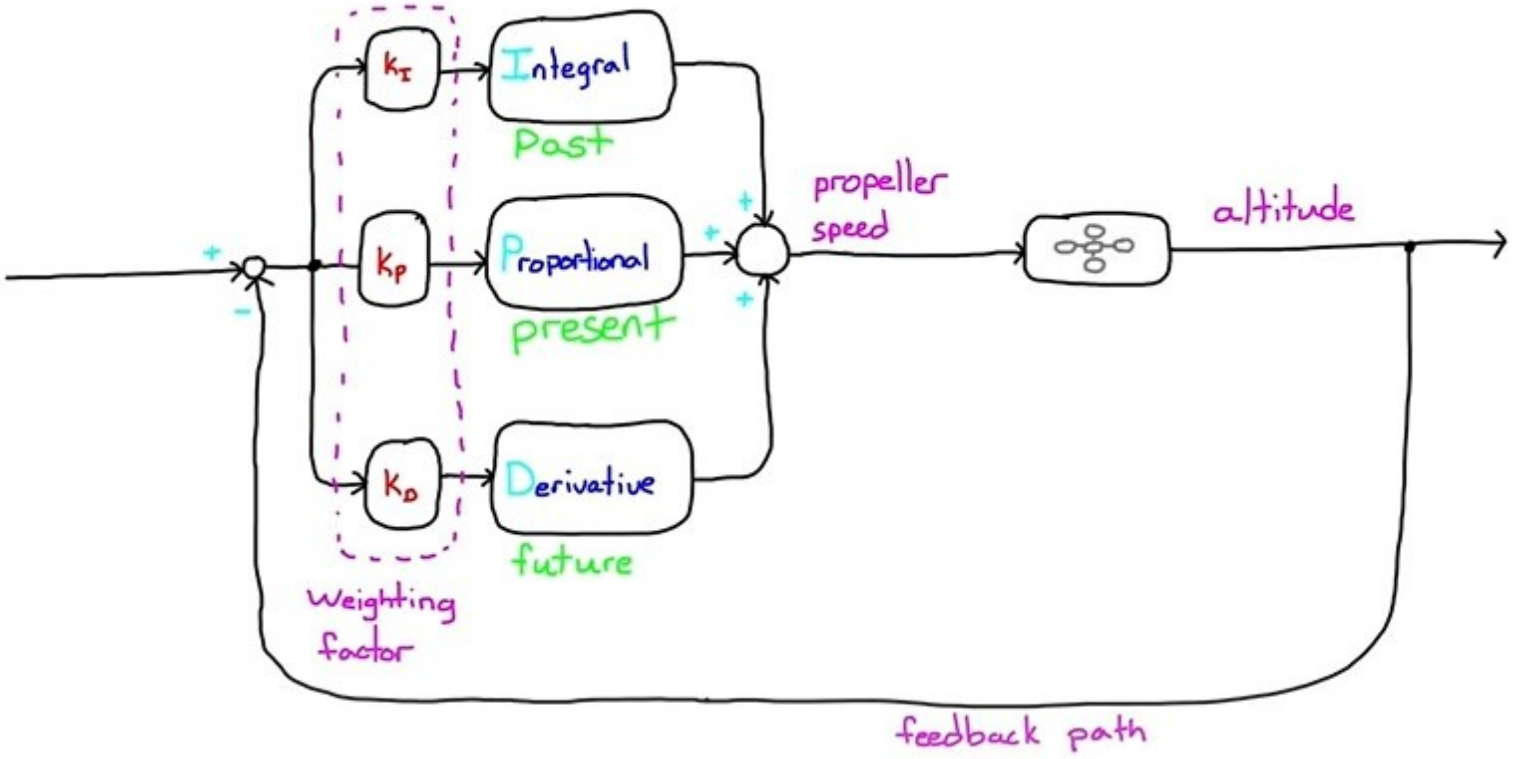
3. Bộ điều khiển vòng kín và bộ điều khiển PID



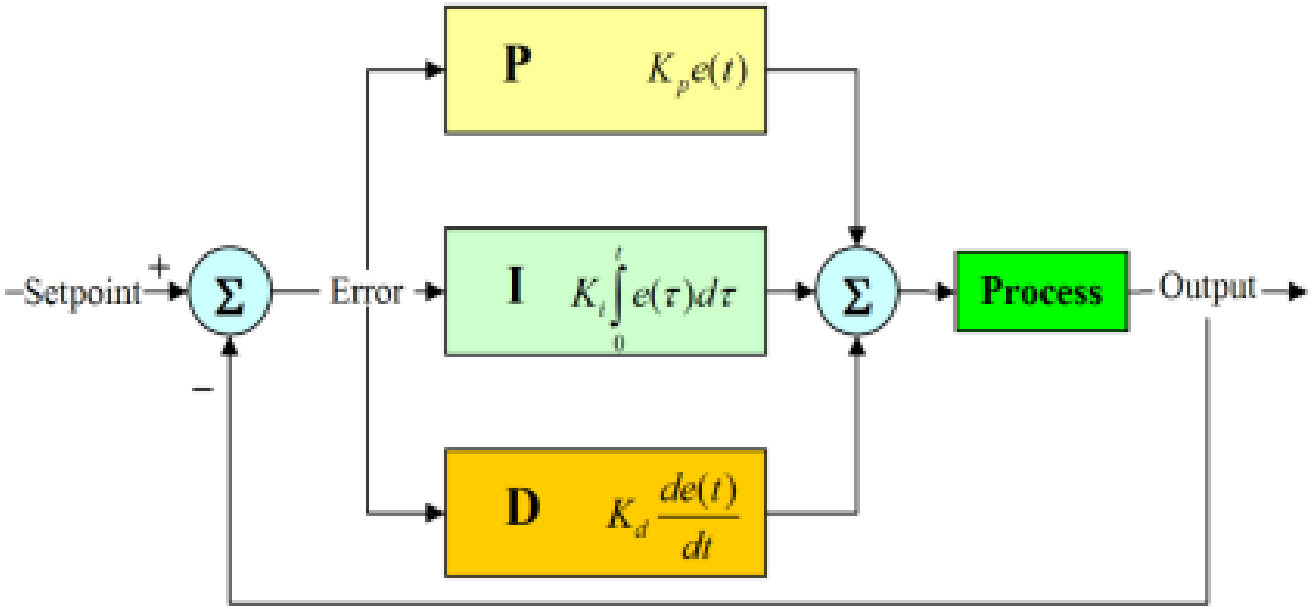
3. Bộ điều khiển vòng kín và bộ điều khiển PID



3. Bộ điều khiển vòng kín và bộ điều khiển PID



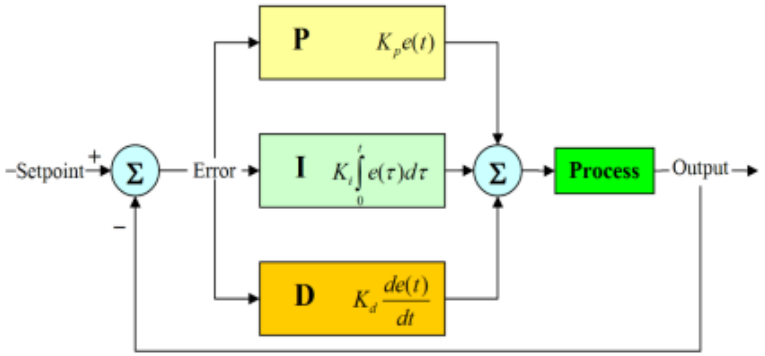
3. Bộ điều khiển vòng kín và bộ điều khiển PID



3. Bộ điều khiển vòng kín và bộ điều khiển PID

Khâu tỉ lệ P (Proportional):

- Tỉ lệ tuyến tính với sai số
- Đổi dấu khi sai số đổi dấu



Ưu điểm:

- Tác động nhanh đơn giản
- Giảm sai số nhanh trong giai đoạn đầu

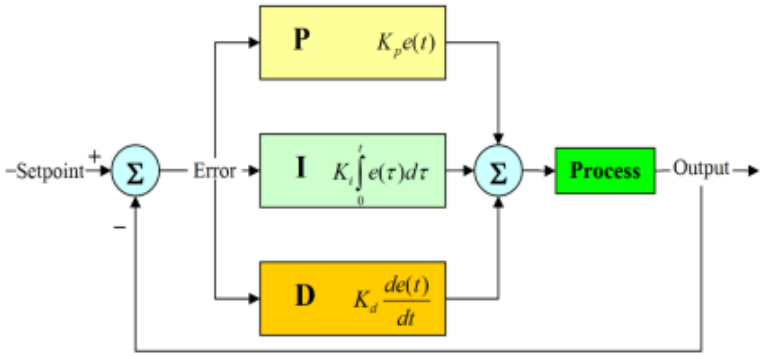
Nhược điểm:

- Dao động lớn nếu hệ số P lớn
- Nhạy cảm với nhiễu đo

3. Bộ điều khiển vòng kín và bộ điều khiển PID

Khâu tích phân I (Integral):

- Còn được gọi là khâu sai số tích lũy
- Tăng/giảm tín hiệu điều khiển khi sai số dương/âm



Ưu điểm:

- Triệt tiêu sai số ở trạng thái xác lập
- Tác động bất kể sai số nhỏ hay lớn

Nhược điểm:

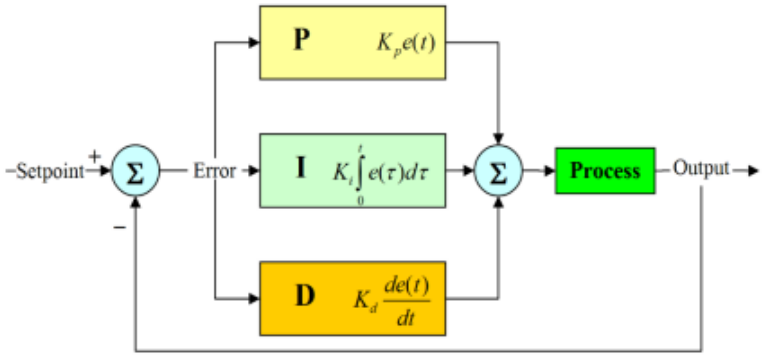
- Chậm xác lập
- Hệ số I quá lớn có thể làm xấu đặc tính động học của hệ thống, dao động đảo chiều liên tục, mất ổn định

3. Bộ điều khiển vòng kín và bộ điều khiển PID

Khâu vi phân D (Derivative):

- Dự đoán đầu ra của

quá trình



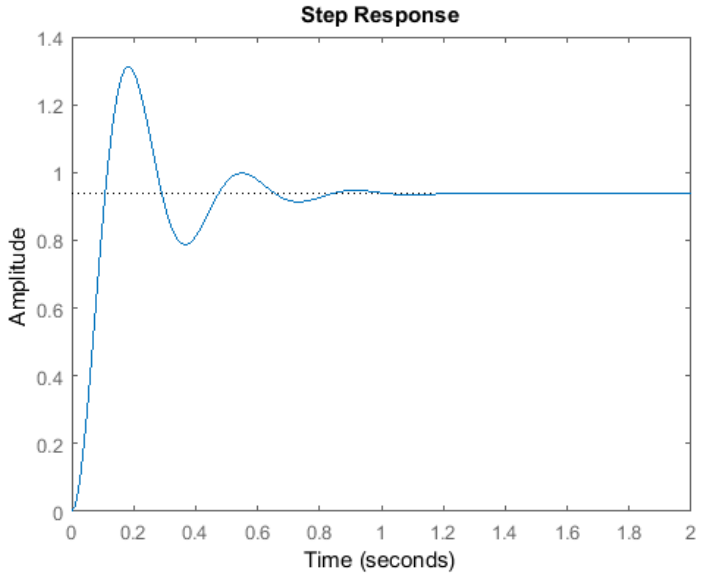
Ưu điểm:

- Ổn định hệ thống
- Tăng tốc độ đáp ứng của hệ thống

Nhược điểm:

- Sai số xác lập
- Có khoảng trễ so với thay đổi của hệ thống

3. Bộ điều khiển vòng kín và bộ điều khiển PID



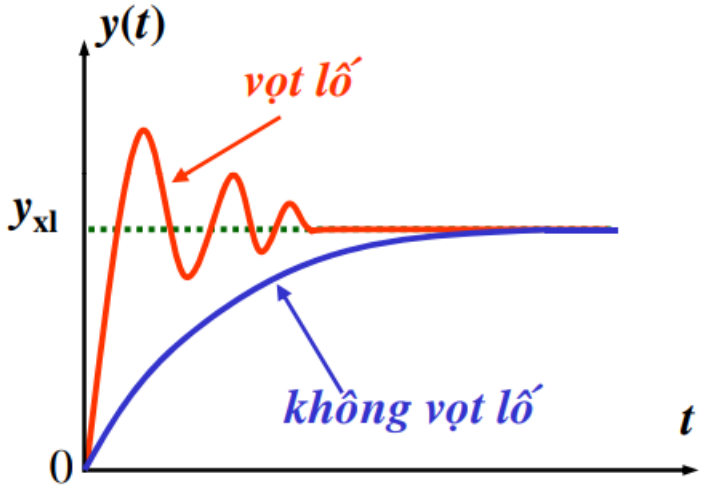
	Thời gian lên	Vọt lố	Thời gian quá độ	Sai số xác lập	Độ ổn định
K_p	Giảm	Tăng	Thay đổi nhỏ	Giảm	Giảm
K_i	Giảm	Tăng	Tăng	Loại bỏ	Giảm
K_d	Thay đổi nhỏ	Giảm	Giảm	Không tác động	Cải thiện

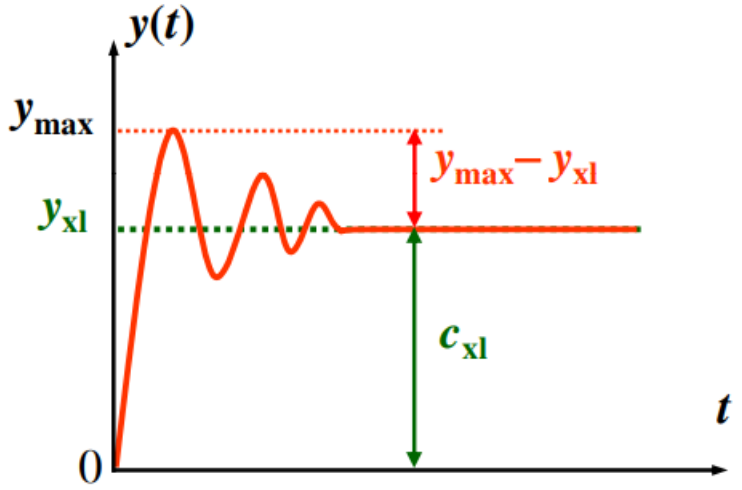
3. Bộ điều khiển vòng kín và bộ điều khiển PID

- Độ vọt lố :

- Percentage overshoot (POT):

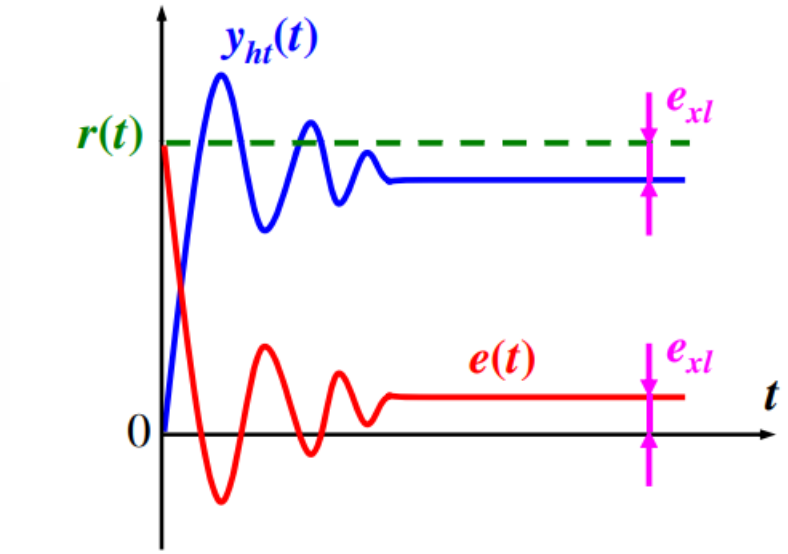
$$POT = \frac{y_{\max} - y_{xl}}{y_{xl}}$$





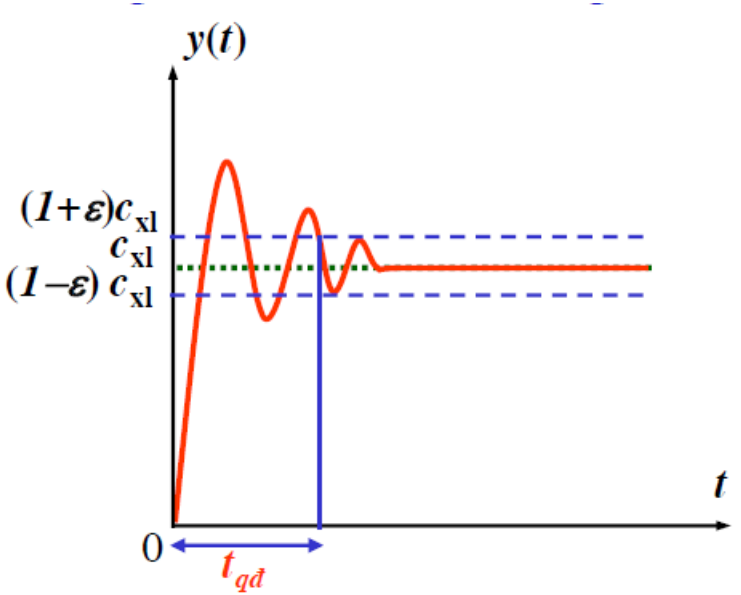
3. Bộ điều khiển vòng kín và bộ điều khiển PID

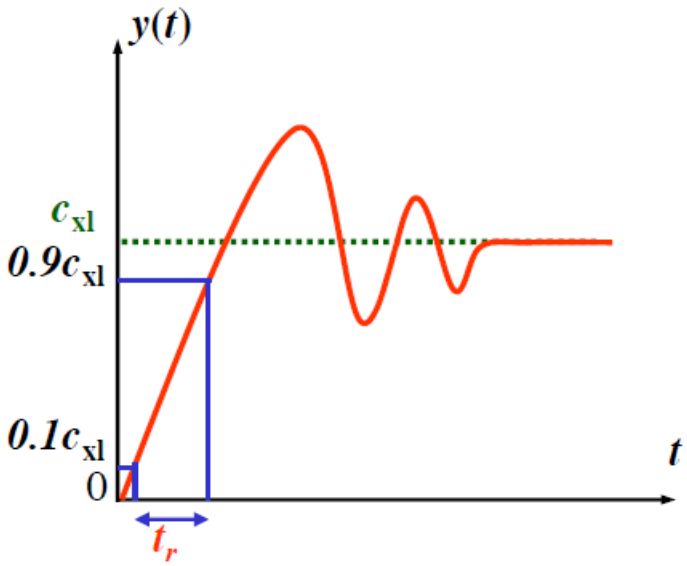
- Sai số xác lập:



3. Bộ điều khiển vòng kín và bộ điều khiển PID

- Thời gian quá độ và thời gian lên:





3. Bộ điều khiển vòng kín và bộ điều khiển PID

- Công thức bộ điều khiển PID số:

$$u(k) = u_P(k) + u_I(k) + u_D(k)$$

Với u(k) là giá trị đầu ra của bộ pid tại thời điểm k
e(k) là giá trị sai số điều khiển tại thời điểm k
e(k) là giá trị sai số điều khiển tại thời điểm k-1 (trước thời điểm k 1 mẫu)
Ts là thời gian lấy mẫu
Kp, Ki, Kd là trọng số các khâu P, I và D của bộ điều khiển

$$u(k) = u_P(k) + u_I(k) + u_D(k) = e(k) \times G(z)$$

3. Bộ điều khiển vòng kín và bộ điều khiển PID

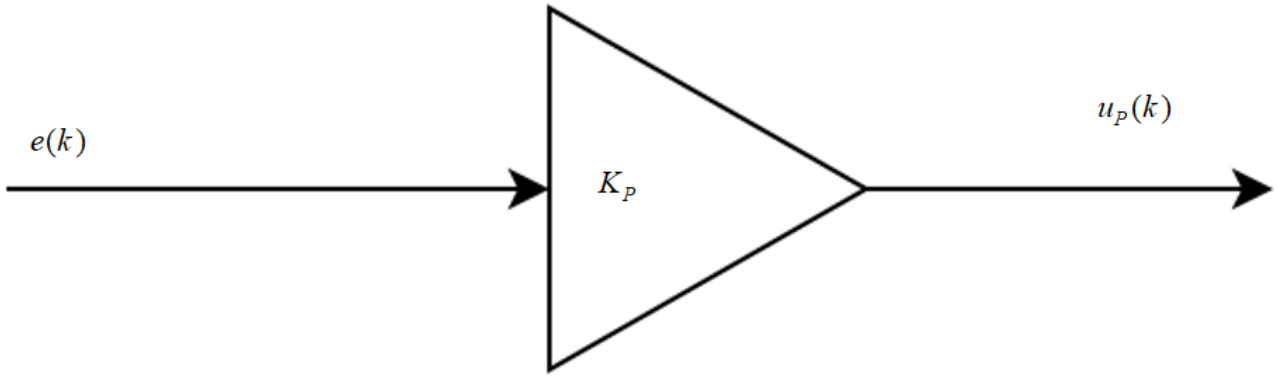
- Ta có công thức với từng khâu như sau:

$$u_P(k) = K_P e(k)$$

$$u_I(k) = K_I \frac{T_s}{2}\left[e(k) + e(k-1)\right] + u_I(k-1)$$

$$u_D(k) = K_D \frac{2}{Ts}[e(k) - e(k-1)] - u_D(k-1)$$

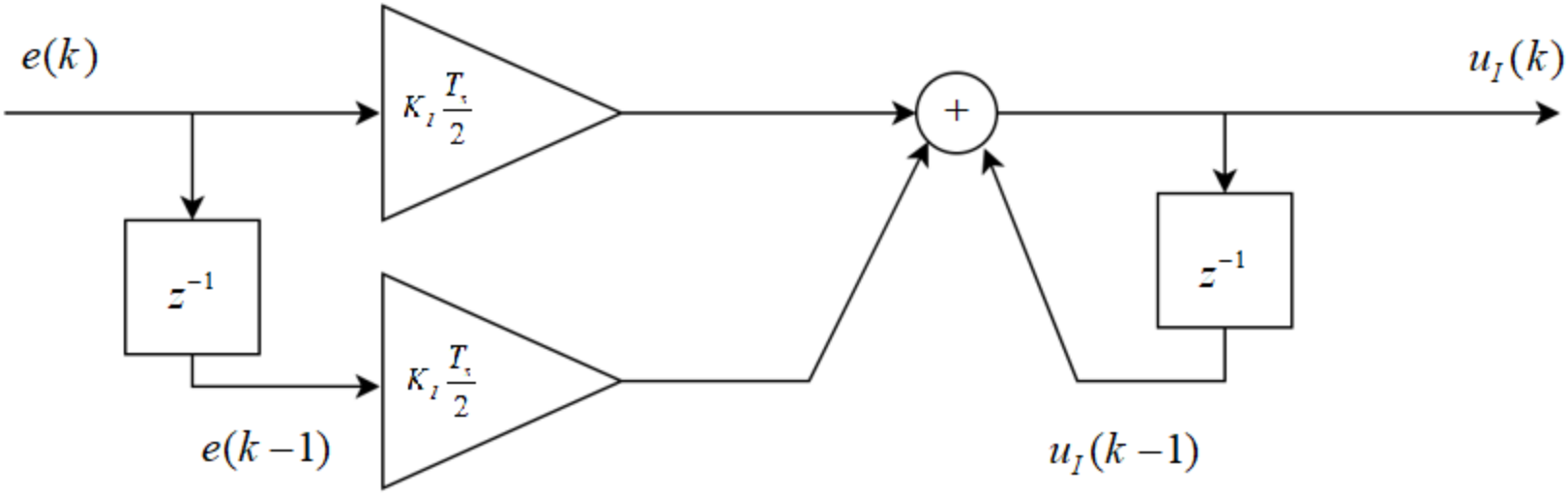
3. Bộ điều khiển vòng kín và bộ điều khiển PID



Với mỗi mẫu e(k):

$$u_P(k) = K_P e(k)$$

3. Bộ điều khiển vòng kín và bộ điều khiển PID



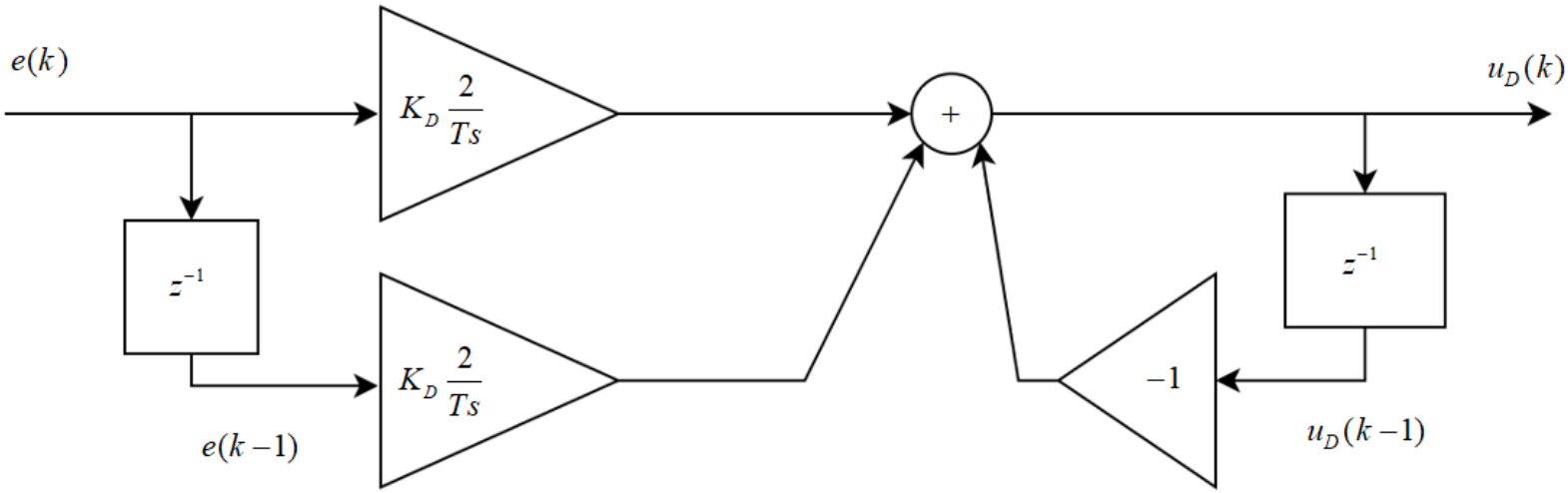
Với mỗi mẫu e(k):

$$u_I(k) = K_I \frac{T_s}{2}\left[e(k) + e(k-1)\right] + u_I(k-1)$$

$$e(k-1) = e(k)$$

$$u_I(k-1) = u_I(k)$$

3. Bộ điều khiển vòng kín và bộ điều khiển PID



Với mỗi mẫu e(k):

$$u_D(k) = K_D \frac{2}{Ts}[e(k) - e(k-1)] - u_D(k-1)$$

$$e(k-1) = e(k)$$

$$u_D(k-1) = u_D(k)$$

3. Bộ điều khiển vòng kín và bộ điều khiển PID

- Phương pháp "tunning" các trọng số Kp, Ki, Kd cho bộ điều khiển PID:

1. Đặt Ki, và Kd, bằng 0, tăng dần Kp đến giá trị hệ số khuếch đại giới hạn Kgn (tức là hệ số khuếch đại làm cho hệ thống ở biên giới ổn định)

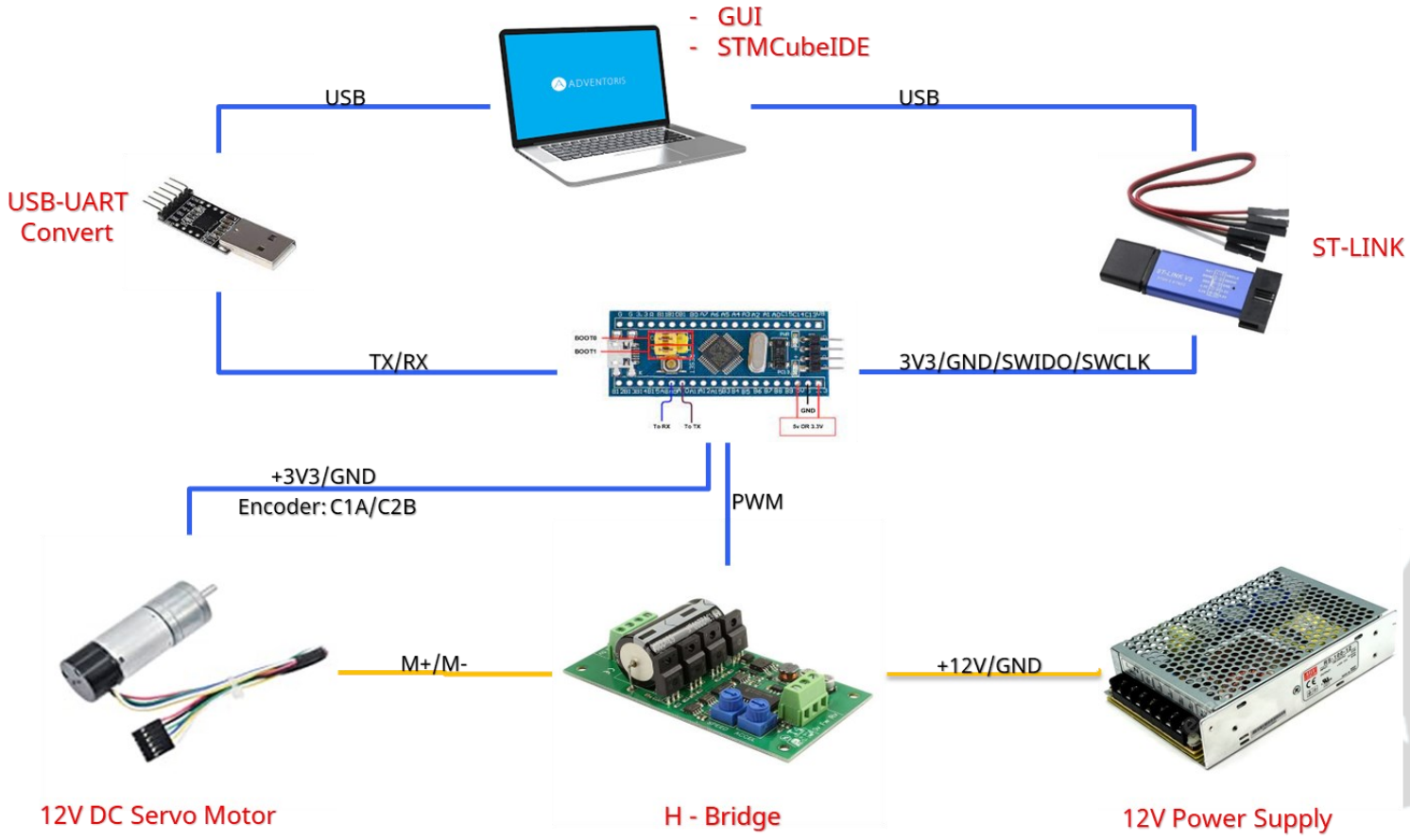
2. Đặt Kp~Kgh/2 (hoặc chỉnh Kp để POT ~ 25%)

3. Tăng dần Ki, đến khi sai số xác lập bị triệt tiêu trong thời gian đủ nhanh (Chú ý Ki, quá lớn có thể làm hệ thống mất ổn định).

4. Tăng dần Kd để giảm độ vọt lố và thời gian xác lập (chú ý Kd quá lớn có thể làm có hệ thống vọt lố trở lại)

5. Tinh chỉnh Kp, Ki, Kd để đáp ứng đạt yêu cầu

4. Thiết kế bộ điều khiển PID cho động cơ DC



4. Thiết kế bộ điều khiển PID cho động cơ DC

• Firmware:

Gồm các phần:

- Serial (quản lý giao tiếp UART)
- Pid (Bộ điều khiển pid)
- Motor (Động cơ DC)

4. Thiết kế bộ điều khiển PID cho động cơ DC

- State machine

NONE: Trạng thái mặc định/nghỉ của hệ thống

SPID: Nhận thông số Kp, Ki, Kd

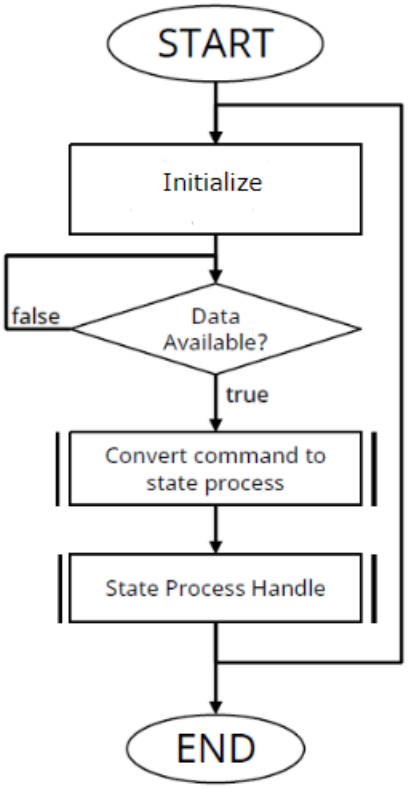
VTUN: Gắn Setpoint vận tốc và điều khiển vận tốc theo thông số bộ pid

PTUN: Gắn Setpoint vị trí và điều khiển vị trí theo thông số bộ pid

STOP: Dừng VTUN và PTUN, reset bộ nhớ

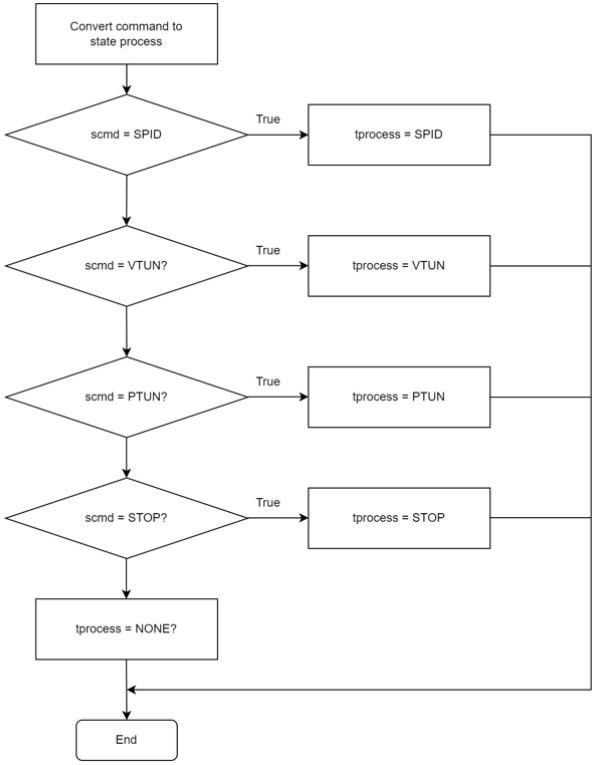
4. Thiết kế bộ điều khiển PID cho động cơ DC

- Firmware flowchart: Main program



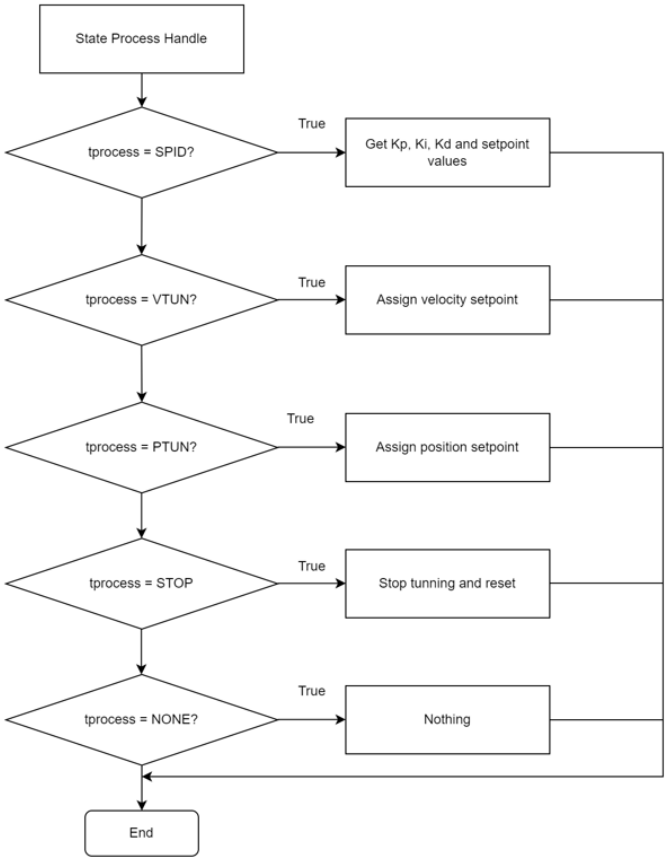
4. Thiết kế bộ điều khiển PID cho động cơ DC

- Convert command to state process



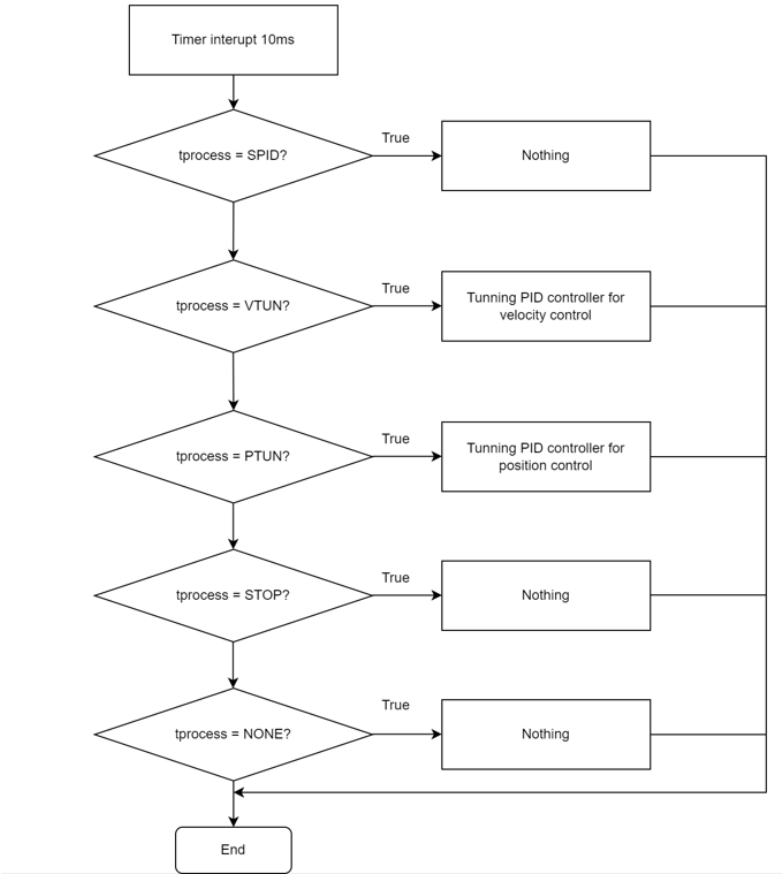
4. Thiết kế bộ điều khiển PID cho động cơ DC

- State Process Handle

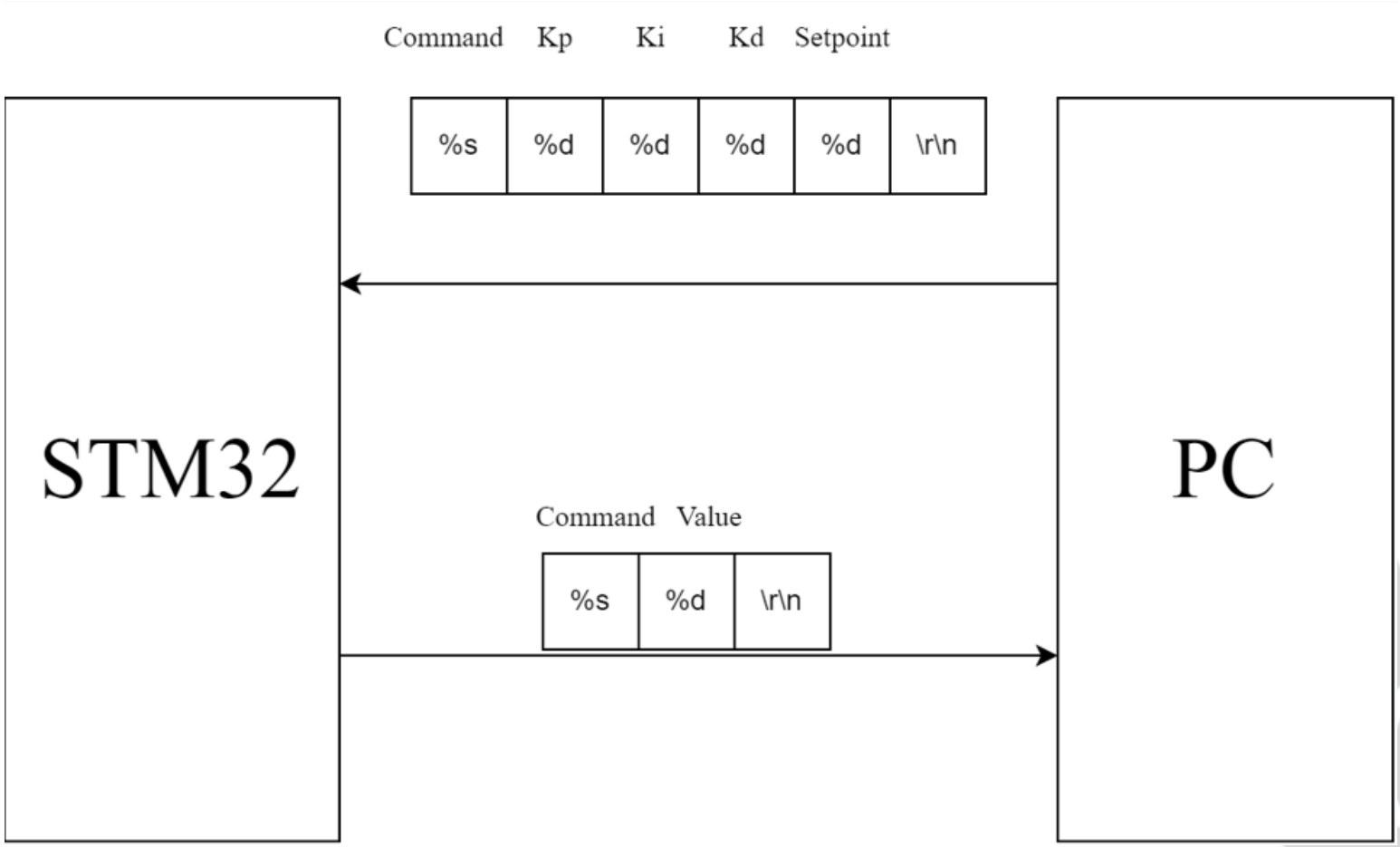


4. Thiết kế bộ điều khiển PID cho động cơ DC

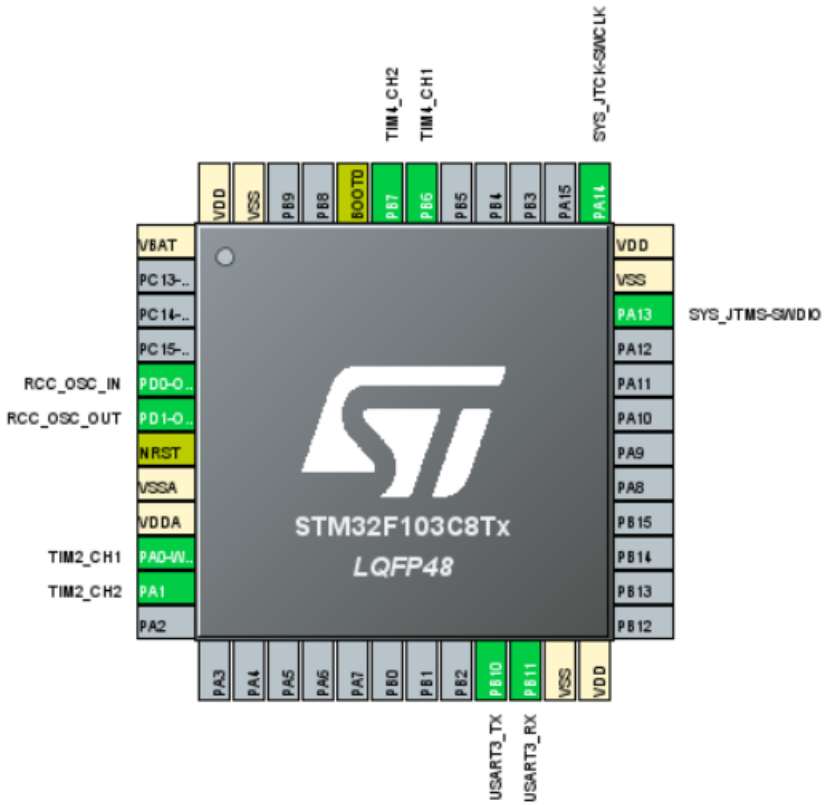
- Firmware flowchart: Timer interrupt 10ms



4. Thiết kế bộ điều khiển PID cho động cơ DC

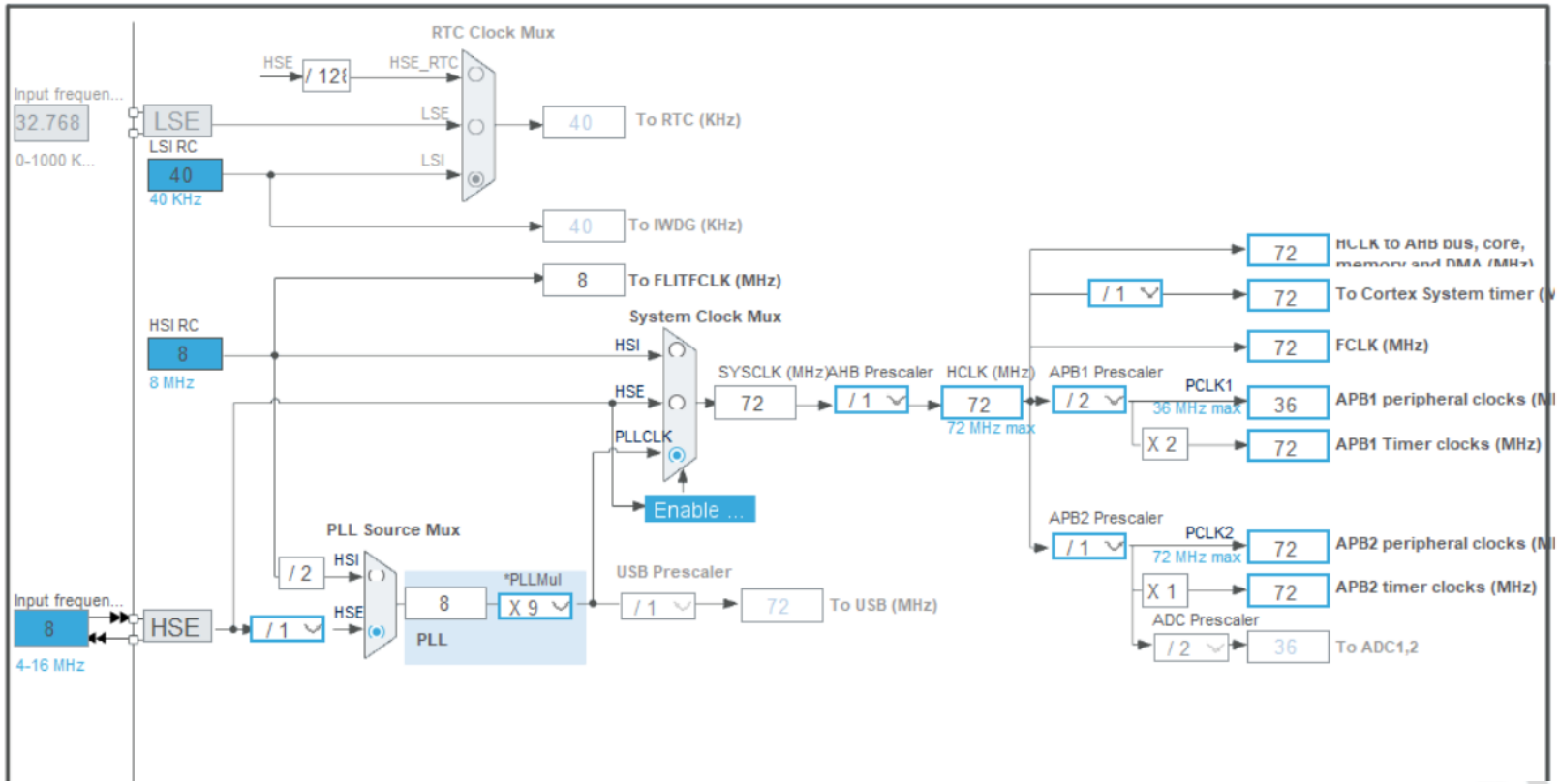


4. Thiết kế bộ điều khiển PID cho động cơ DC



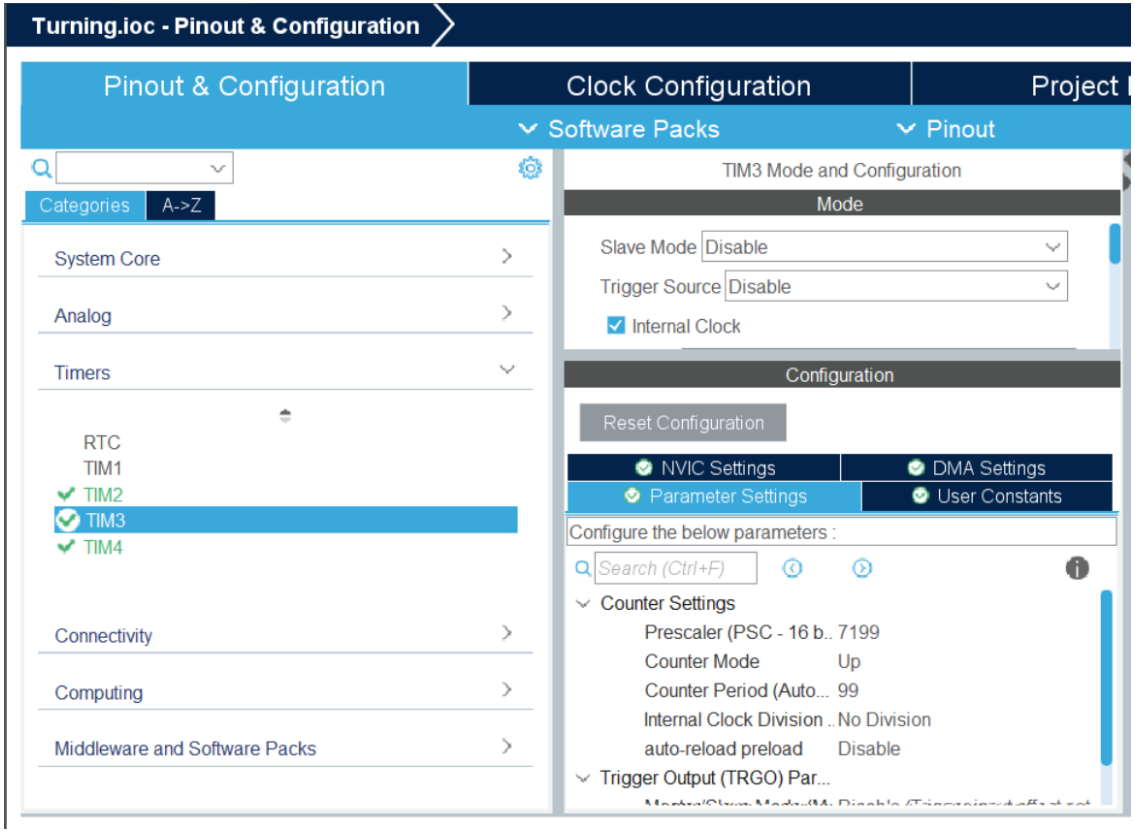
PIN	Name and function	Connect with
PA0	Channel 1-Timer 2 PWM	H Bridge IN/PWM
PA1	Channel 2-Timer 2 PWM	H Bridge IN/PWM
PB6	Channel 1-Timer 4 Encoder A	Encoder A
PB7	Channel 1-Timer 4 Encoder B	Encoder B
PB10	UART 3 TX	RX
PB11	UART 3 RX	TX

4. Thiết kế bộ điều khiển PID cho động cơ DC



4. Thiết kế bộ điều khiển PID cho động cơ DC

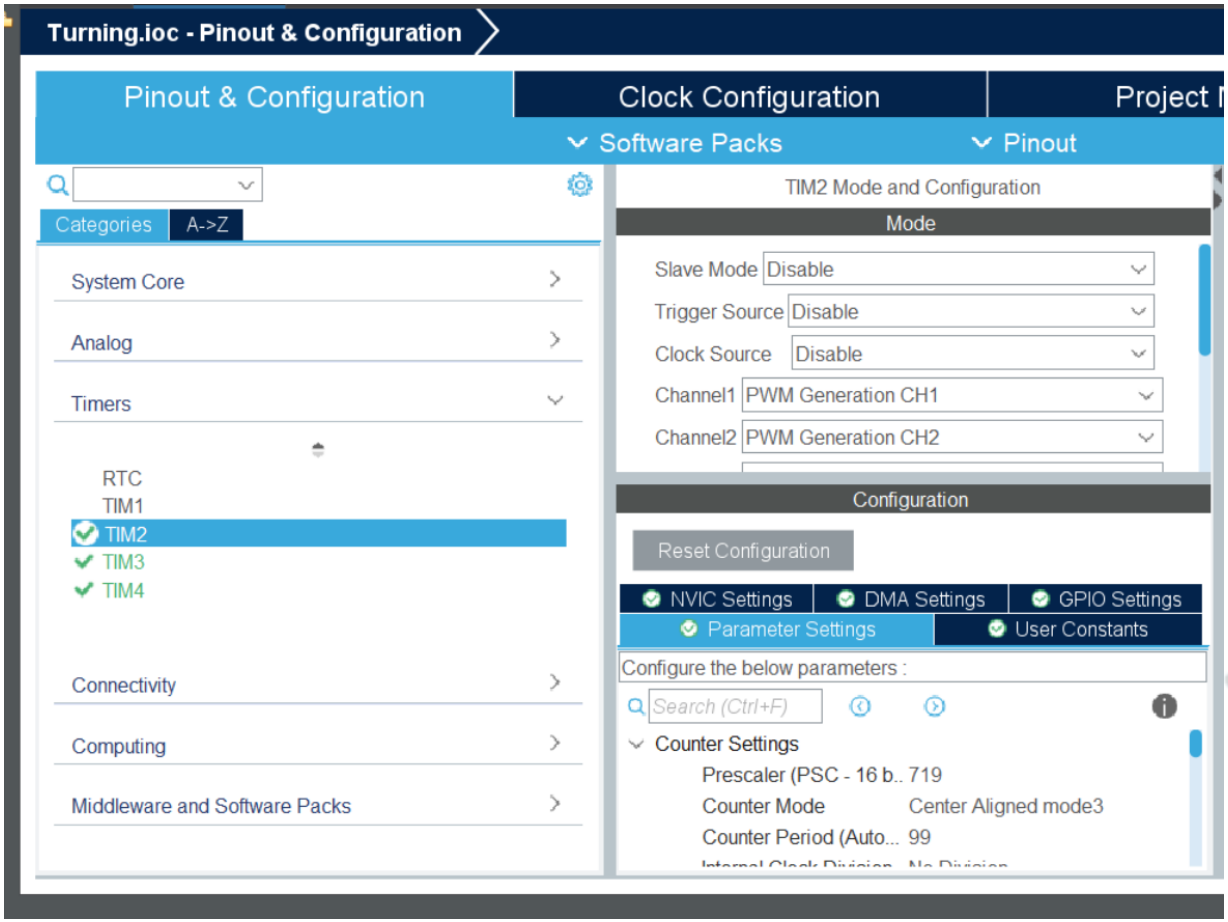
- Config timer interupt 10ms:



NVIC Interrupt Table	Enabled	Preemption Priority	
TIM3 global interrupt	☑	2	0

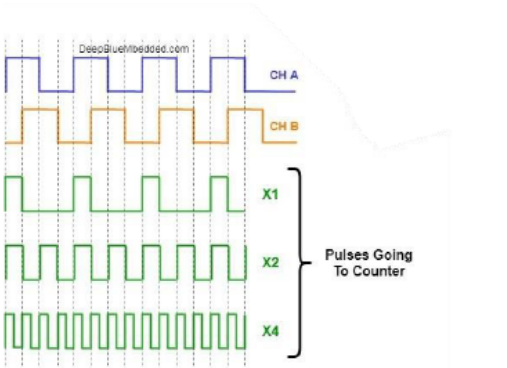
4. Thiết kế bộ điều khiển PID cho động cơ DC

- Config PWM 1KHz:

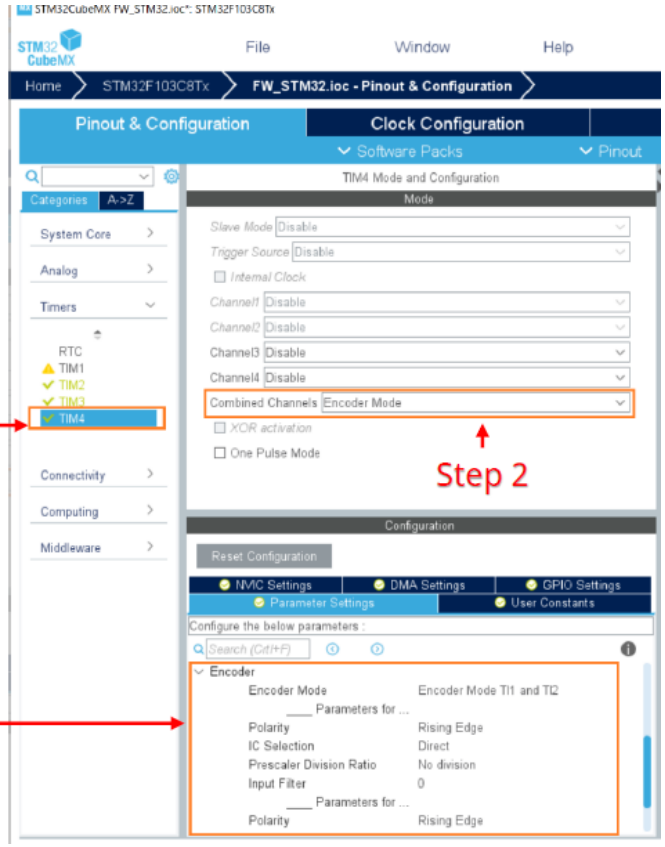


4. Thiết kế bộ điều khiển PID cho động cơ DC

- Config encoder counter:



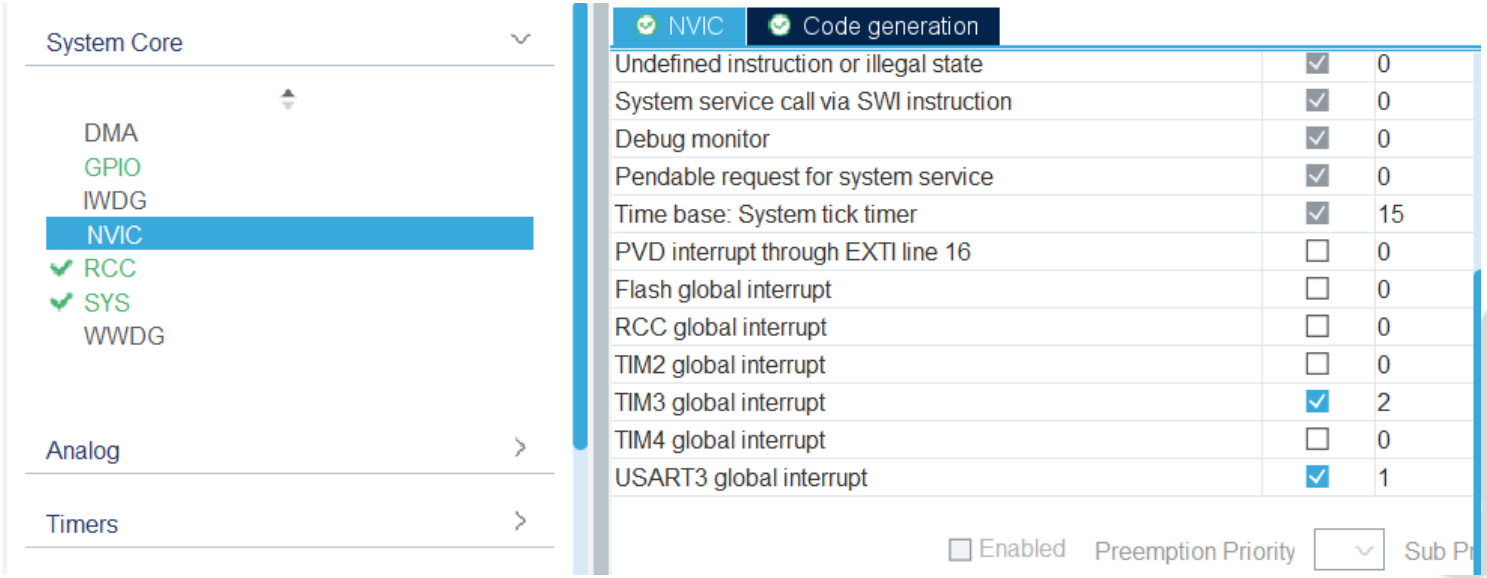
Config Encoder X4
⇒ Mode: **Encoder Mode TI1 and TI2**
⇒ Polarity: **Rising Edge**



4. Thiết kế bộ điều khiển PID cho động cơ DC

- UART interrupt:

UART interrupt's Priority > Timer interrupt's Priority



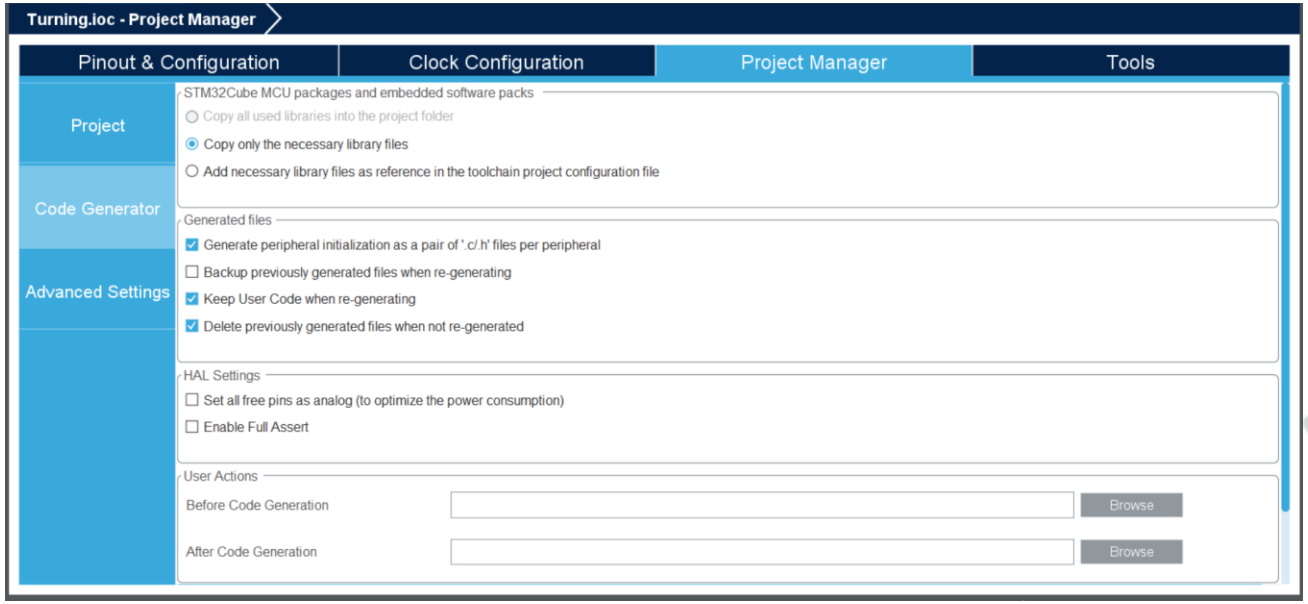
4. Thiết kế bộ điều khiển PID cho động cơ DC

- Project manager:

Tab Code Generator

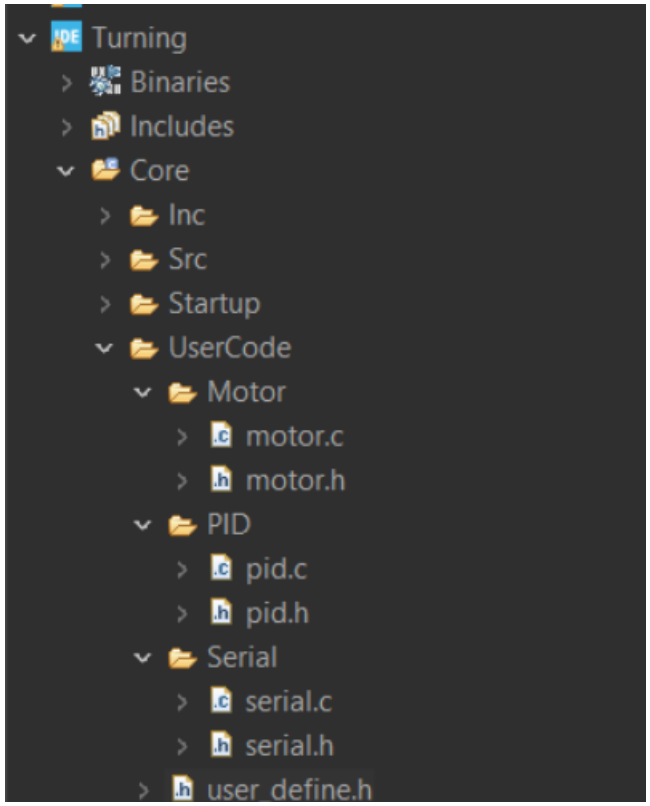
Tick "Copy only the necessary library files"

Click "**GENERATE CODE**" to generate C project



4. Thiết kế bộ điều khiển PID cho động cơ DC

- Create file .c, .h



4. Thiết kế bộ điều khiển PID cho động cơ DC

- Create file

pid.c, pid.h ->Functions for control PID

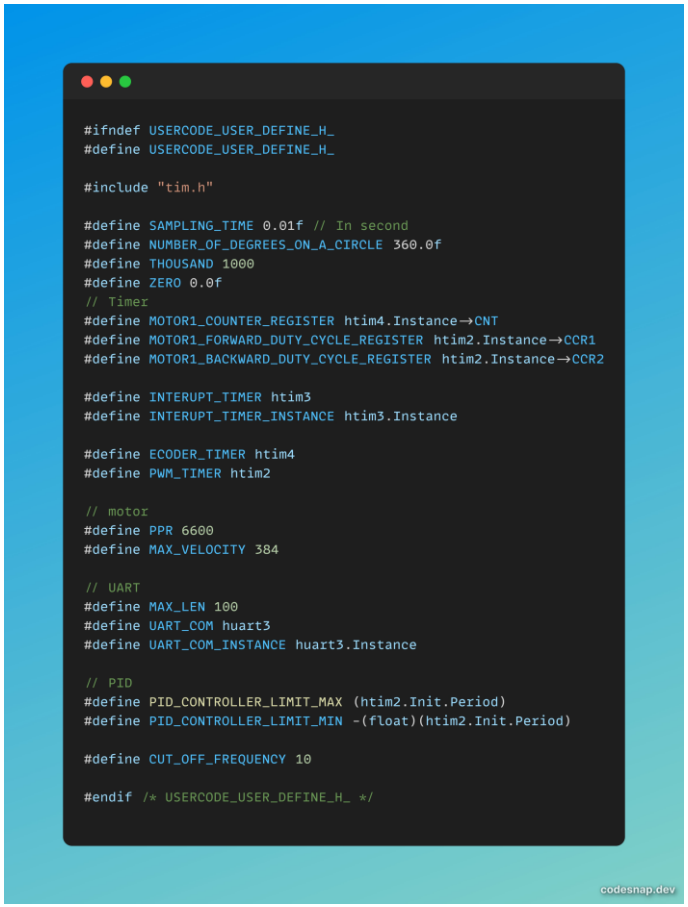
motor.c, motor.h ->Functions for control motor

serial.c, serial.h ->Functions for serial interface

user_define.h ->Macro

4. Thiết kế bộ điều khiển PID cho động cơ DC

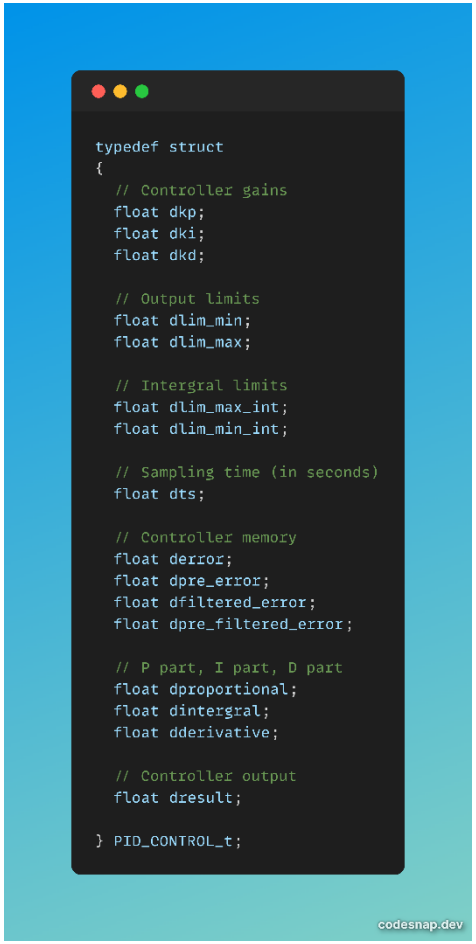
- user_define.h



```
#ifndef USERCODE_USER_DEFINE_H_
#define USERCODE_USER_DEFINE_H_

#include "tim.h"

#define SAMPLING_TIME 0.01f // In second
#define NUMBER_OF_DEGREES_ON_A_CIRCLE 360.0f
#define THOUSAND 1000
#define ZERO 0.0f
// Timer
#define MOTOR1_COUNTER_REGISTER htim4.Instance->CNT
#define MOTOR1_FORWARD_DUTY_CYCLE_REGISTER htim2.Instance->CCR1
#define MOTOR1_BACKWARD_DUTY_CYCLE_REGISTER htim2.Instance->CCR2

#define INTERUPT_TIMER htim3
#define INTERUPT_TIMER_INSTANCE htim3.Instance

#define ECODER_TIMER htim4
#define PWM_TIMER htim2

// motor
#define PPR 6600
#define MAX_VELOCITY 384

// UART
#define MAX_LEN 100
#define UART_COM huart3
#define UART_COM_INSTANCE huart3.Instance

// PID
#define PID_CONTROLLER_LIMIT_MAX (htim2.Init.Period)
#define PID_CONTROLLER_LIMIT_MIN -(float)(htim2.Init.Period)

#define CUT_OFF_FREQUENCY 10

#endif /* USERCODE_USER_DEFINE_H_ */
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- Structure:



```c
typedef struct
{
  // Controller gains
  float dkp;
  float dki;
  float dkd;

  // Output limits
  float dlim_min;
  float dlim_max;

  // Intergral limits
  float dlim_max_int;
  float dlim_min_int;

  // Sampling time (in seconds)
  float dts;

  // Controller memory
  float derror;
  float dpre_error;
  float dfiltered_error;
  float dpre_filtered_error;

  // P part, I part, D part
  float dproportional;
  float dintergral;
  float dderivative;

  // Controller output
  float dresult;

} PID_CONTROL_t;
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- Structure:

```
typedef struct
{
  uint32_t ipulse_per_round;
  int16_t icounter;
  float dvelocity;
  float dposition;
  float dreference_velocity;
  float dreference_position;
} Motor_t;
```

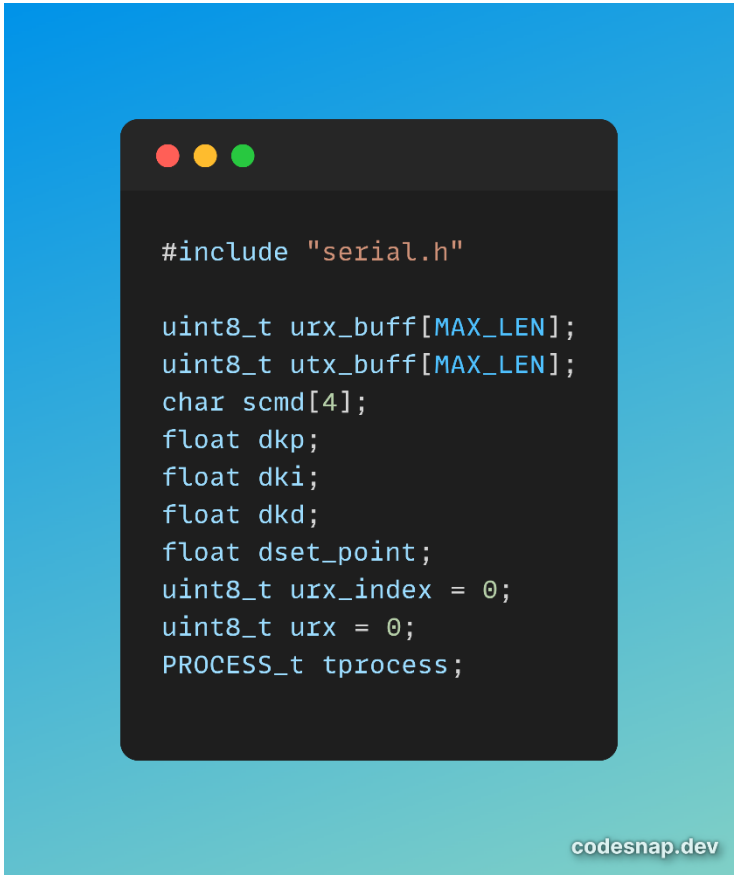
codesnap.dev

```
typedef enum
{
  NONE = 0,
  SPID,
  VTUN,
  PTUN,
  STOP,
} PROCESS_t;
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

- serial.c

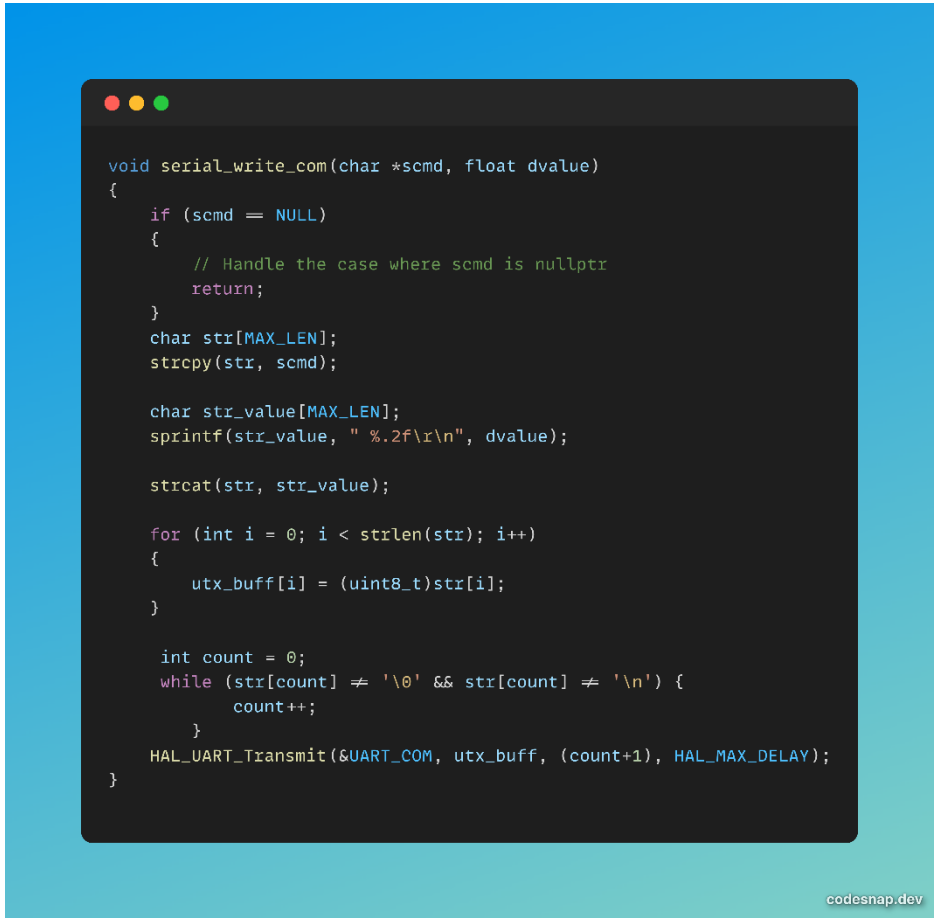


4. Thiết kế bộ điều khiển PID cho động cơ DC

- serial.c

4. Thiết kế bộ điều khiển PID cho động cơ DC

- serial.c



```c
void serial_write_com(char *scmd, float dvalue)
{
    if (scmd == NULL)
    {
        // Handle the case where scmd is nullptr
        return;
    }
    char str[MAX_LEN];
    strcpy(str, scmd);

    char str_value[MAX_LEN];
    sprintf(str_value, " %.2f\r\n", dvalue);

    strcat(str, str_value);

    for (int i = 0; i < strlen(str); i++)
    {
        utx_buff[i] = (uint8_t)str[i];
    }

     int count = 0;
     while (str[count] != '\0' && str[count] != '\n') {
            count++;
        }
    HAL_UART_Transmit(&UART_COM, utx_buff, (count+1), HAL_MAX_DELAY);
}
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- serial.c

```c
void serial_handle(uint8_t *ubuff)
{
    if (ubuff == NULL)
    {
        // Handle the case where ubuff is nullptr
        return;
    }
    char str[MAX_LEN];
    snprintf(str, sizeof(str), "%s", ubuff);
    sscanf(str, "%s %f %f %f %f", scmd, &dkp, &dki, &dkd, &dset_point);
    HAL_UART_Transmit(&UART_COM, ubuff, urx_index, HAL_MAX_DELAY);
    urx_index = 0;
      if (StrCompare(scmd, (uint8_t*)"SPID", 4))
        {
          tprocess = SPID;
        }
        else if (StrCompare(scmd, (uint8_t*)"VTUN", 4))
        {
          tprocess = VTUN;
        }
        else if (StrCompare(scmd, (uint8_t*)"PTUN", 4))
        {
          tprocess = PTUN;
        }
        else if (StrCompare(scmd, (uint8_t*)"STOP", 4))
        {
          tprocess = STOP;
        }
        else
        {
          tprocess = NONE;
        }
}
```

codesnap.dev

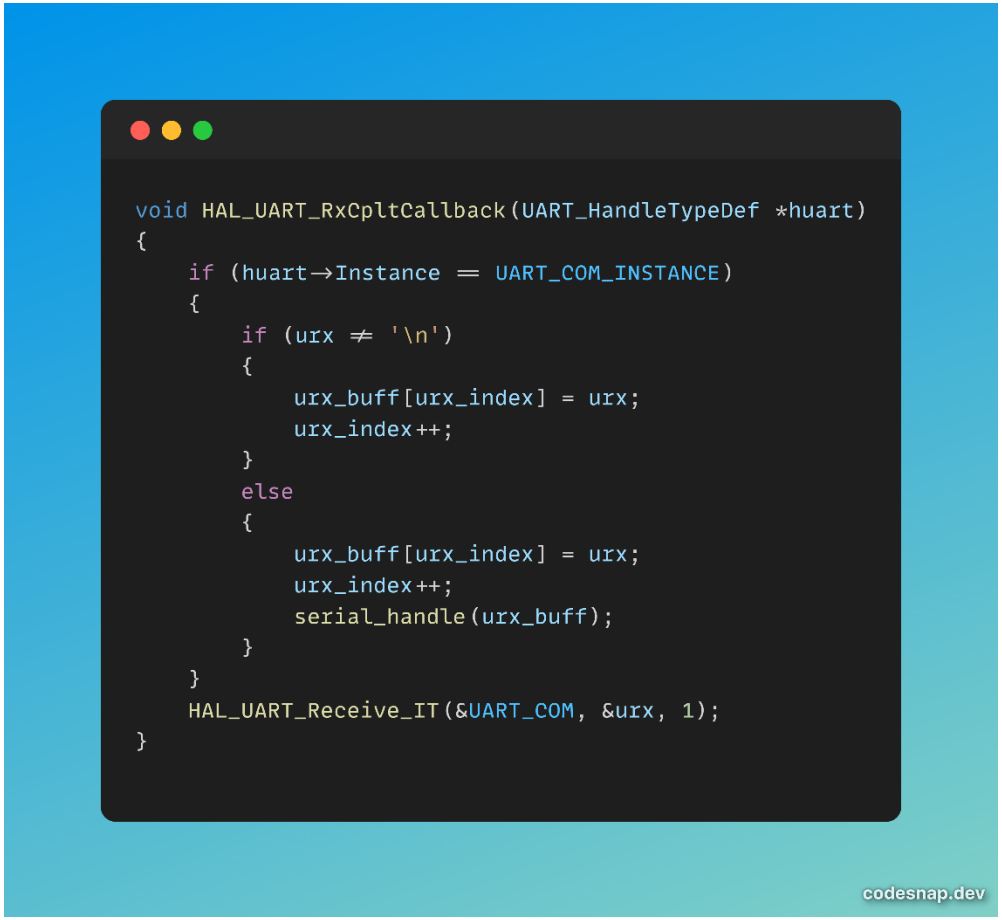
4. Thiết kế bộ điều khiển PID cho động cơ DC

- serial.c

```
bool StrCompare(char *pBuff, uint8_t *pSample, uint8_t nSize)
{
    for (int i = 0; i < nSize; i++)
    {
        if(pBuff[i] != pSample[i])
        {
            return false;
        }
    }
    return true;
}
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- serial.c



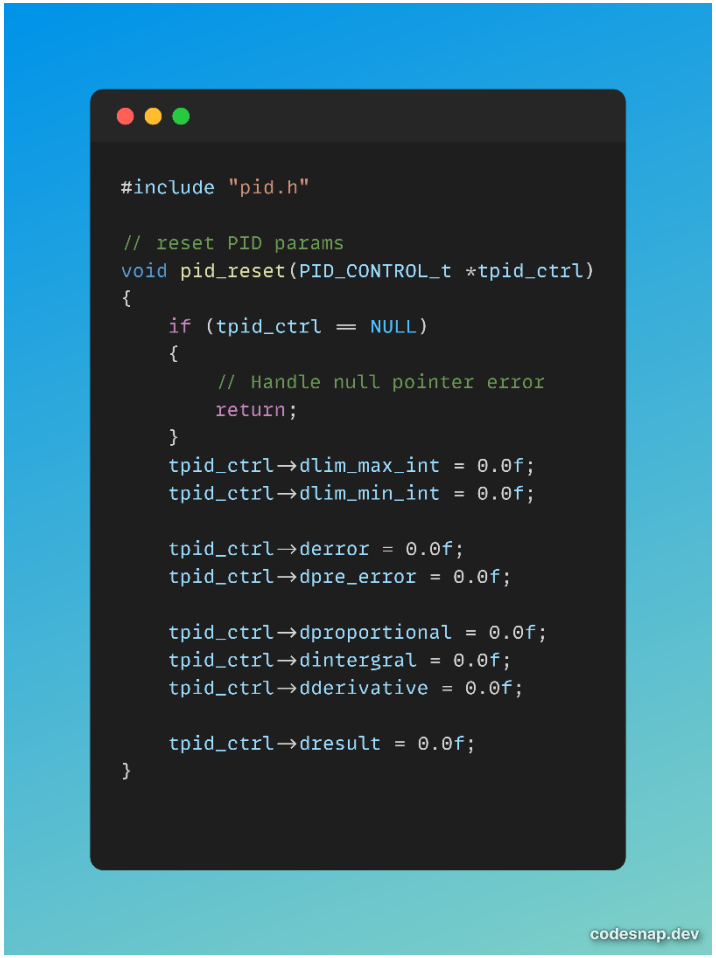
4. Thiết kế bộ điều khiển PID cho động cơ DC

- serial.h

```h
#ifndef INC_SERIAL_H_
#define INC_SERIAL_H_

#include "../UserCode/user_define.h"
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "usart.h"

typedef enum
{
  NONE = 0,
  SPID,
  VTUN,
  PTUN,
  STOP,
} PROCESS_t;

void serial_init(void);
void serial_write_com(char *ucmd, float dvalue);
void serial_handle(uint8_t *ubuff);
bool StrCompare(char *pBuff, uint8_t *pSample, uint8_t nSize);
#endif /* INC_SERIAL_H_ */
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- pid.c



4. Thiết kế bộ điều khiển PID cho động cơ DC

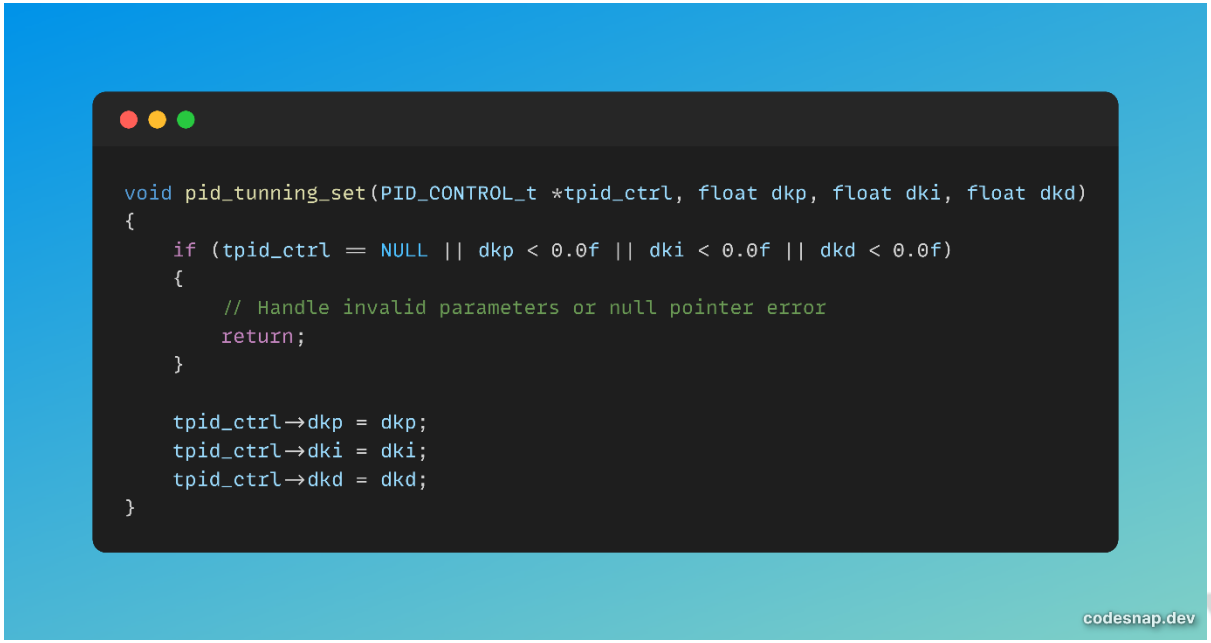
- pid.c

```c
// init PID
void pid_init(PID_CONTROL_t *tpid_ctrl, float dkp, float dki, float dkd, float dlimit_max, float dlimit_min, float dts)
{
    if (tpid_ctrl == NULL || dkp < 0.0f || dki < 0.0f || dkd < 0.0f || dts < 0.0f || dlimit_max < dlimit_min)
    {
        // Handle invalid parameters or null pointer error
        return;
    }
    pid_reset(tpid_ctrl);
    tpid_ctrl->dkp = dkp;
    tpid_ctrl->dki = dki;
    tpid_ctrl->dkd = dkd;

    tpid_ctrl->dlim_max = dlimit_max;
    tpid_ctrl->dlim_min = dlimit_min;

    tpid_ctrl->dts = dts;
}
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

- pid.c



4. Thiết kế bộ điều khiển PID cho động cơ DC

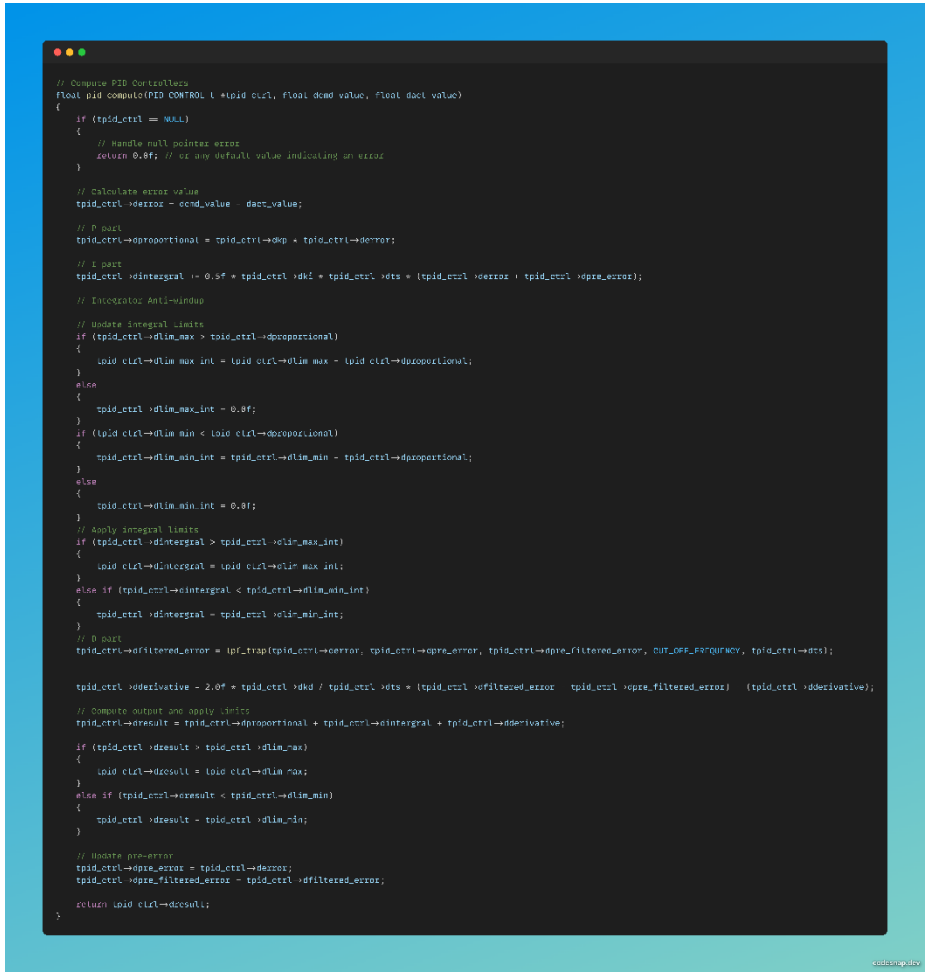
- pid.c

```c
float lpf_trap(float draw_signal_value, float dpre_raw_signal_value, float dpre_filtered_value, float dfc, float dts)
{
    float dfiltered_value = 0.0f;
    float da1 = 0.0f;
    float db0 = 0.0f;
    float db1 = 0.0f;
    float dwc = 0.0f;

    if (dfc < 0.0f || dts < 0.0f)
    {
        return dfiltered_value;
    }

    dwc = dfc * 2 * 3.141592f; // rad/s
    da1 = (2.0f - dwc * dts) / (2.0f + dwc * dts);
    db0 = (dwc * dts) / (2.0f + dwc * dts);
    db1 = db0;
    dfiltered_value = da1 * dpre_filtered_value + db0 * draw_signal_value + db1 * dpre_raw_signal_value;

    return dfiltered_value;
}
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

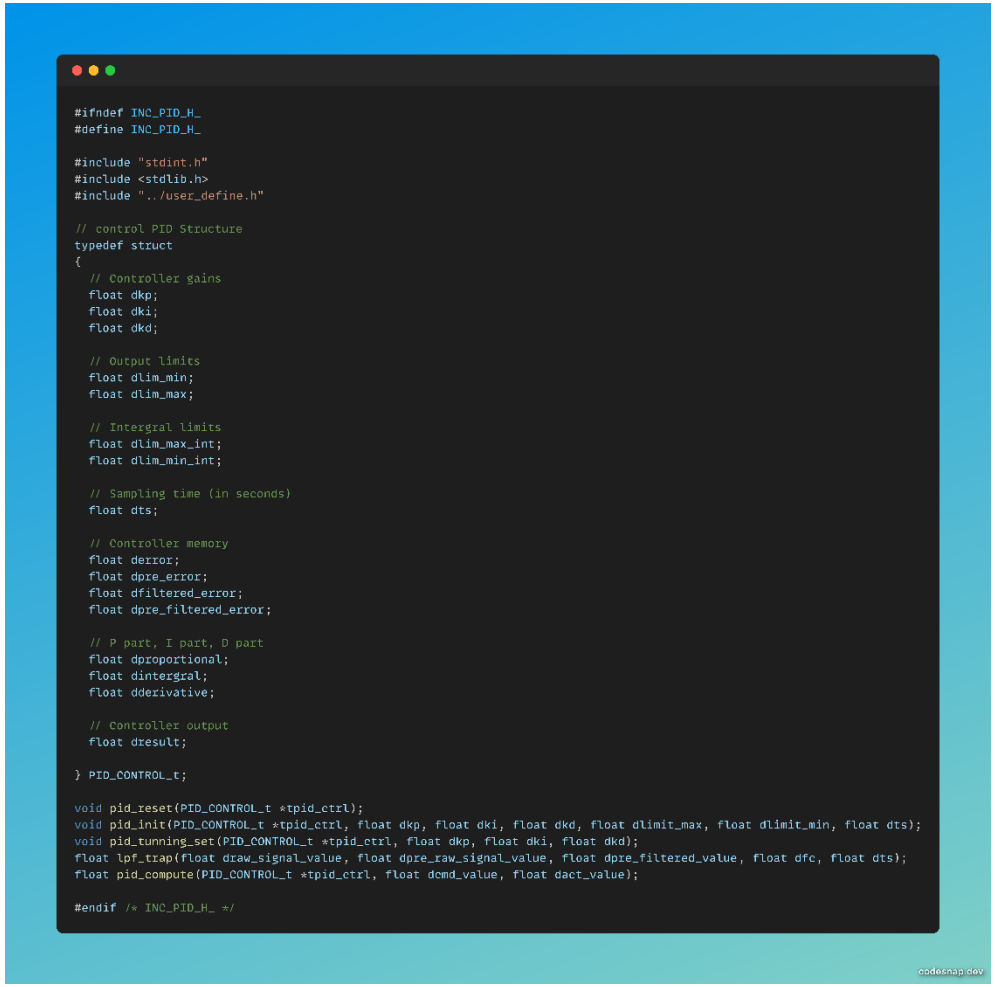
- pid.c



```c
// Compute PID Controllers
float pid_compute(PID_CONTROL_t *pid_ctrl, float demd_value, float dact_value)
{
    if (tpid_ctrl == NULL)
    {
        // Handle null pointer error
        return 0.0f; // or any default value indicating an error
    }

    // Calculate error value
    tpid_ctrl->derror = demd_value - dact_value;

    // P part
    tpid_ctrl->dproportional = tpid_ctrl->dkp * tpid_ctrl->derror;

    // I part
    tpid_ctrl->dintergral += 0.5f * tpid_ctrl->dki * tpid_ctrl->dts * (tpid_ctrl->derror + tpid_ctrl->dpre_error);

    // Integrator Anti-windup

    // Update integral limits
    if (tpid_ctrl->dlim_max > tpid_ctrl->dproportional)
    {
        tpid_ctrl->dlim_max_int = tpid_ctrl->dlim_max - tpid_ctrl->dproportional;
    }
    else
    {
        tpid_ctrl->dlim_max_int = 0.0f;
    }
    if (tpid_ctrl->dlim_min < tpid_ctrl->dproportional)
    {
        tpid_ctrl->dlim_min_int = tpid_ctrl->dlim_min - tpid_ctrl->dproportional;
    }
    else
    {
        tpid_ctrl->dlim_min_int = 0.0f;
    }
    // Apply integral limits
    if (tpid_ctrl->dintergral > tpid_ctrl->dlim_max_int)
    {
        tpid_ctrl->dintergral = tpid_ctrl->dlim_max_int;
    }
    else if (tpid_ctrl->dintergral < tpid_ctrl->dlim_min_int)
    {
        tpid_ctrl->dintergral = tpid_ctrl->dlim_min_int;
    }
    // D part
    tpid_ctrl->dfiltered_error = lpf_trap(tpid_ctrl->derror, tpid_ctrl->dpre_error, tpid_ctrl->dpre_filtered_error, CUT_OFF_FREQUENCY, tpid_ctrl->dts);


    tpid_ctrl->dderivative = 2.0f * tpid_ctrl->dkd / tpid_ctrl->dts * (tpid_ctrl->dfiltered_error - tpid_ctrl->dpre_filtered_error) - (tpid_ctrl->dderivative);

    // Compute output and apply limits
    tpid_ctrl->dresult = tpid_ctrl->dproportional + tpid_ctrl->dintergral + tpid_ctrl->dderivative;

    if (tpid_ctrl->dresult > tpid_ctrl->dlim_max)
    {
        tpid_ctrl->dresult = tpid_ctrl->dlim_max;
    }
    else if (tpid_ctrl->dresult < tpid_ctrl->dlim_min)
    {
        tpid_ctrl->dresult = tpid_ctrl->dlim_min;
    }

    // Update pre-error
    tpid_ctrl->dpre_error = tpid_ctrl->derror;
    tpid_ctrl->dpre_filtered_error = tpid_ctrl->dfiltered_error;

    return tpid_ctrl->dresult;
}
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- pid.h



```h
#ifndef INC_PID_H_
#define INC_PID_H_

#include "stdint.h"
#include <stdlib.h>
#include "../user_define.h"

// control PID Structure
typedef struct
{
  // Controller gains
  float dkp;
  float dki;
  float dkd;

  // Output limits
  float dlim_min;
  float dlim_max;

  // Intergral limits
  float dlim_max_int;
  float dlim_min_int;

  // Sampling time (in seconds)
  float dts;

  // Controller memory
  float derror;
  float dpre_error;
  float dfiltered_error;
  float dpre_filtered_error;

  // P part, I part, D part
  float dproportional;
  float dintergral;
  float dderivative;

  // Controller output
  float dresult;

} PID_CONTROL_t;

void pid_reset(PID_CONTROL_t *tpid_ctrl);
void pid_init(PID_CONTROL_t *tpid_ctrl, float dkp, float dki, float dkd, float dlimit_max, float dlimit_min, float dts);
void pid_tunning_set(PID_CONTROL_t *tpid_ctrl, float dkp, float dki, float dkd);
float lpf_trap(float draw_signal_value, float dpre_raw_signal_value, float dpre_filtered_value, float dfc, float dts);
float pid_compute(PID_CONTROL_t *tpid_ctrl, float domd_value, float dact_value);

#endif /* INC_PID_H_ */
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- motor.c

```c
#include "motor.h"

void motor_init(Motor_t *tmotor, uint32_t ipulse)
{
    if (tmotor == NULL)
    {
        // Handle null pointer error
        return;
    }

    motor_reset(tmotor);
    tmotor->ipulse_per_round = ipulse;
    HAL_TIM_Base_Start_IT(&INTERUPT_TIMER);
    HAL_TIM_Encoder_Start(&ECODER_TIMER, TIM_CHANNEL_1);
    HAL_TIM_Encoder_Start(&ECODER_TIMER, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start(&PWM_TIMER, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&PWM_TIMER, TIM_CHANNEL_2);
}
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

- motor.c

```c
#include "motor.h"

void motor_init(Motor_t *tmotor, uint32_t ipulse)
{
    if (tmotor == NULL)
    {
        // Handle null pointer error
        return;
    }

    motor_reset(tmotor);
    tmotor->ipulse_per_round = ipulse;
    HAL_TIM_Base_Start_IT(&INTERUPT_TIMER);
    HAL_TIM_Encoder_Start(&ECODER_TIMER, TIM_CHANNEL_1);
    HAL_TIM_Encoder_Start(&ECODER_TIMER, TIM_CHANNEL_2);
    HAL_TIM_PWM_Start(&PWM_TIMER, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&PWM_TIMER, TIM_CHANNEL_2);
}
```

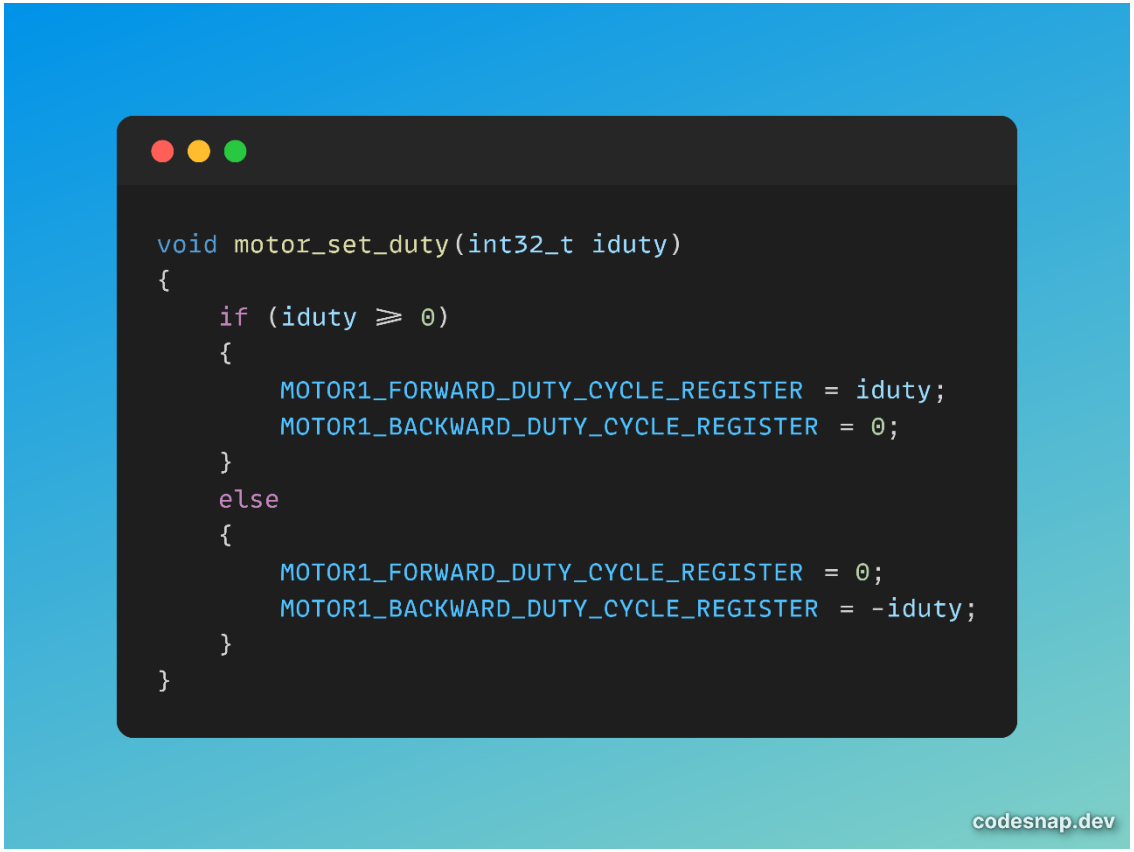
codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

- motor.c

4. Thiết kế bộ điều khiển PID cho động cơ DC

- motor.c



```
void motor_set_duty(int32_t iduty)
{
    if (iduty >= 0)
    {
        MOTOR1_FORWARD_DUTY_CYCLE_REGISTER = iduty;
        MOTOR1_BACKWARD_DUTY_CYCLE_REGISTER = 0;
    }
    else
    {
        MOTOR1_FORWARD_DUTY_CYCLE_REGISTER = 0;
        MOTOR1_BACKWARD_DUTY_CYCLE_REGISTER = -iduty;
    }
}
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- motor.c

```c
void motor_read_encoder(Motor_t *tmotor, TIM_HandleTypeDef *htim)
{
    if (tmotor == NULL || htim == NULL)
    {
        // Handle null pointer error
        return;
    }
    tmotor->icounter = htim->Instance->CNT;
    tmotor->dvelocity = (float)tmotor->icounter / (float)tmotor->ipulse_per_round * NUMBER_OF_DEGREES_ON_A_CIRCLE / SAMPLING_TIME;
    tmotor->dposition += (float)tmotor->icounter / (float)tmotor->ipulse_per_round * NUMBER_OF_DEGREES_ON_A_CIRCLE;
    htim->Instance->CNT = 0;
}
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

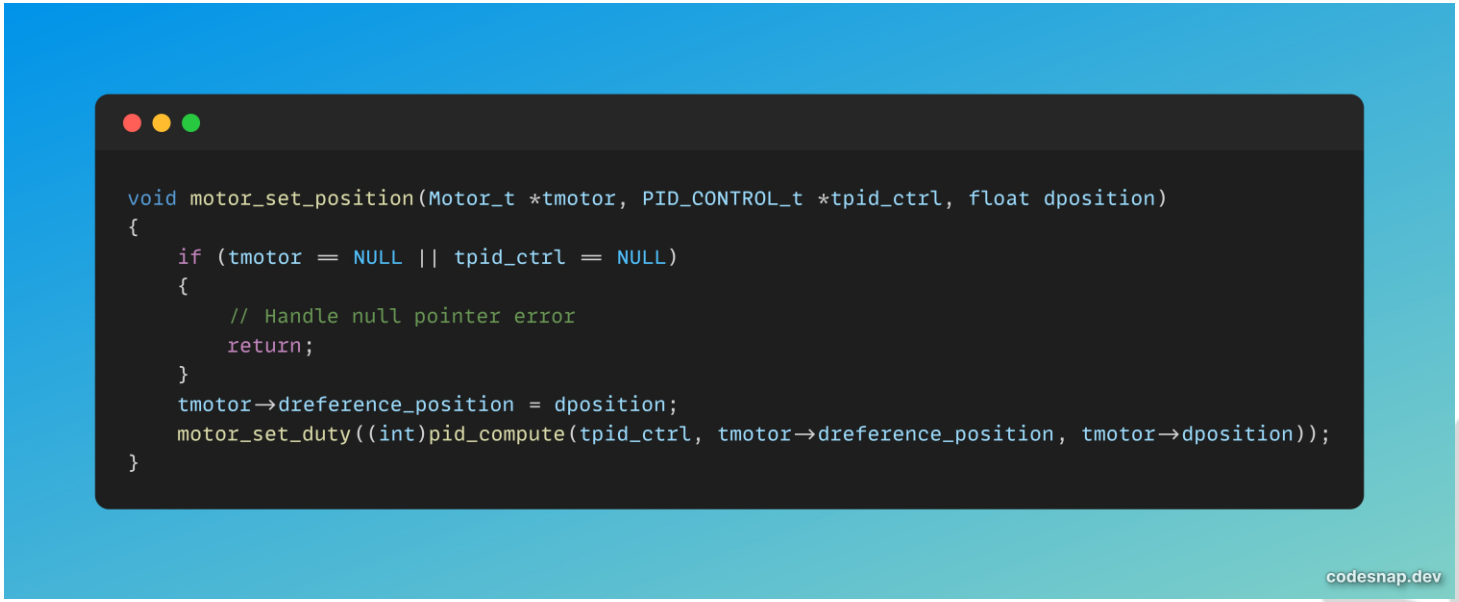
- motor.c

```c
void motor_set_velocity(Motor_t *tmotor, PID_CONTROL_t *tpid_ctrl, float dvelocity)
{
    if (tmotor == NULL || tpid_ctrl == NULL)
    {
        // Handle null pointer error
        return;
    }

    tmotor->dreference_velocity = dvelocity;
    motor_set_duty((int)pid_compute(tpid_ctrl, tmotor->dreference_velocity, tmotor->dvelocity));
}
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

- motor.c



```
void motor_set_position(Motor_t *tmotor, PID_CONTROL_t *tpid_ctrl, float dposition)
{
    if (tmotor == NULL || tpid_ctrl == NULL)
    {
        // Handle null pointer error
        return;
    }
    tmotor->dreference_position = dposition;
    motor_set_duty((int)pid_compute(tpid_ctrl, tmotor->dreference_position, tmotor->dposition));
}
```

4. Thiết kế bộ điều khiển PID cho động cơ DC

- motor.h

```h
#ifndef INC_MOTOR_H_
#define INC_MOTOR_H_

#include <stdint.h>
#include <stdlib.h>
#include <math.h>
#include "../PID/pid.h"
#include "../user_define.h"
#include "tim.h"

typedef struct
{
  uint32_t ipulse_per_round;
  int16_t icounter;
  float dvelocity;
  float dposition;
  float dreference_velocity;
  float dreference_position;
} Motor_t;

void motor_init(Motor_t *tmotor, uint32_t ipulse);
void motor_reset(Motor_t *tmotor);
void motor_set_duty(int32_t iduty);
void motor_read_encoder(Motor_t *tmotor, TIM_HandleTypeDef *htim);
void motor_set_velocity(Motor_t *tmotor, PID_CONTROL_t *tpid_ctrl, float dvelocity);
void motor_set_position(Motor_t *tmotor, PID_CONTROL_t *tpid_ctrl, float dposition);

#endif /* INC_MOTOR_H_ */
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

- main.c

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "stdio.h"
#include "string.h"
#include "../UserCode/Motor/motor.h"
#include "../UserCode/PID/pid.h"
#include "../UserCode/Serial/serial.h"
/* USER CODE END Includes */
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

- main.c

```
/* USER CODE BEGIN PV */
Motor_t tmotor;
PID_CONTROL_t tpid;
extern PROCESS_t tprocess;

extern uint8_t urx_buff[MAX_LEN];
extern uint8_t utx_buff[MAX_LEN];
extern char scmd[4];
extern float dkp;
extern float dki;
extern float dkd;
extern float dset_point;
/* USER CODE END PV */
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

- main.c

```
/* USER CODE BEGIN 2 */
motor_init(&tmotor, PPR);
pid_init(&tpid, ZERO, ZERO, ZERO, PID_CONTROLLER_LIMIT_MAX, PID_CONTROLLER_LIMIT_MIN, SAMPLING_TIME);
serial_init();
/* USER CODE END 2 */
```

codesnap.dev

4. Thiết kế bộ điều khiển PID cho động cơ DC

4. Thiết kế bộ điều khiển PID cho động cơ DC

```c
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if (htim->Instance == INTERUPT_TIMER_INSTANCE)
  {
    switch (tprocess)
    {
    case NONE:
      break;
    case SPID:
      break;
    case VTUN:
      motor_read_encoder(&tmotor, &ECODER_TIMER);
      motor_set_velocity(&tmotor, &tpid, dset_point);
      serial_write_com(scmd, tmotor.dvelocity);
      break;
    case PTUN:
      motor_read_encoder(&tmotor, &ECODER_TIMER);
      motor_set_position(&tmotor, &tpid, dset_point);
      serial_write_com(scmd, tmotor.dposition);
      break;
    case STOP:
      break;
    }
  }
}
/* USER CODE END 4 */
```

codesnap.dev

5. Phụ lục

- Khai triển công thức bộ điều khiển PID số
- Bộ ngăn vọt lố do khâu I
- Bộ lọc thông thấp cho khâu D
- Triển khai bộ lọc và bộ điều khiển theo dạng chính tắc

5. Phụ lục

• Khai triển công thức bộ điều khiển PID số

- Ta có công thức bộ điều khiển PID trong miền thời gian như sau:

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\,\mathrm{d}\tau + K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

5. Phụ lục

• Khai triển công thức bộ điều khiển PID số

- Công thức bộ điều khiển PID trong miền thời gian biểu diễn hệ thống liên tục nên nó không phù hợp với nền tảng vi điều khiển (lấy mẫu và đưa ra tín hiệu điều khiển rời rạc với tần số lấy mẫu nhất định)

5. Phụ lục

• Khai triển công thức bộ điều khiển PID số

- Thay vì chuyển thẳng công thức bộ điều khiển PID liên tục ở miền thời gian sang công thức rời rạc thì ta chọn cách đi từ công thức ở miền thời gian sang miền tần số sau đó mới chuyển sang miền rời rạc

- Để hiểu về lý do thì trước hết ta cần hiểu qua về 3 miền thời gian, tần số và rời rạc

5. Phụ lục

• Miền thời gian t:

- Đây là miền trong đó chúng ta xem xét phản ứng của hệ thống theo thời gian. Trong miền này, chúng ta quan tâm đến các thông số như thời gian đáp ứng, dạng sóng, và dạng đáp ứng tự do (đối với hệ thống một chiều)

5. Phụ lục

• Miền tần số phức s:

- Miền này liên quan đến phân tích các phản ứng của hệ thống dưới góc độ tần số. Trong miền S, chúng ta sử dụng biến đổi Laplace để chuyển đổi các phương trình vi phân thông thường sang phương trình đại số tuyến tính, giúp dễ dàng phân tích và thiết kế hệ thống điều khiển

5. Phụ lục

• Miền rời rạc z:

- Miền z là miền của biến số rời rạc, thường được sử dụng trong xử lý tín hiệu kỹ thuật số. Trong miền này, tín hiệu được biểu diễn bằng các giá trị lấy mẫu tại các điểm cố định trong thời gian. Công thức PID được triển khai trong miền z để điều khiển các hệ thống rời rạc hoặc để thiết kế bộ lọc số.

5. Phụ lục

- Như vậy khi trong miền tần số, ta vẫn đánh giá được đáp ứng của hệ thống một cách liên tục nhưng nó đơn giản hóa các phép tích phân và đạo hàm theo thời gian, thêm vào đó là đáp ứng của hệ thống đối với các dải tần số khác nhau
- Do đó, người ta thường sử dụng công thức trong miền tần số khi xét đến các hệ thống liên tục thay vì miền thời gian

5. Phụ lục

- Quay trở lại công thức bộ điều khiển PID trong miền thời gian

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\,\mathrm{d}\tau + K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

$$=> \frac{u(t)}{e(t)} = g(t) = Kp + Ki\int_0^t d\tau + Kd\,\frac{d}{dt}$$

5. Phụ lục

• Biến đổi Laplace

- Dùng biến đổi Laplace để chuyển từ miền thời gian sang miền tần số.

- Đơn gian hóa biến đổi Laplace:

$$\frac{d}{dt} \longrightarrow s$$

$$\int_0^t d\tau \longrightarrow \frac{1}{s}$$

5. Phụ lục

• Biến đổi Laplace:

- Để hiểu sâu về biến đổi Laplace, tham khảo khóa học sau: LaplaceTranformCourse

5. Phụ lục

$$g(t) = Kp + Ki\int_0^t d\tau + Kd\frac{d}{dt}$$

Công thức trong miền s:

$$\Rightarrow G(s) = Kp + Ki\frac{1}{s} + Kds$$

5. Phụ lục

• Công thức bộ điều khiển PID số

- Xấp xỉ giá trị của phép tích phân và vi phân trong miền rời rạc như sau:

$$s \longrightarrow \frac{2}{T_s}\frac{z-1}{z+1} = \frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}}$$

T_s : chu kỳ lấy mẫu

5. Phụ lục

- Ở đây, phép xấp xỉ sử dụng luật hình thang:Trapezoidal rule (differential equations)
- Các trên Internet thường xử dụng xấp xỉ Euler: Backward Euler method

5. Phụ lục

- Công thức hàm điều khiển PID trong miền z:

$$G(z) = K_P + K_I \frac{Ts}{2} \frac{1+z^{-1}}{1-z^{-1}} + K_D \frac{2}{Ts} \frac{1-z^{-1}}{1+z^{-1}}$$

5. Phụ lục

- Từ công thức rời rạc, ta chuyển đổi sang phương trình sai phân:

$$u(k) \times z^{-1} = u(k-1)$$

Với u(k) là giá trị dời dạc tại thời điểm k và u(k-1) là giá trị dời dạc tại thời điểm k -1(trước thời điểm k 1 lần lấy mẫu)

5. Phụ lục

$$G(z) = K_P + K_I \frac{Ts}{2} \frac{1+z^{-1}}{1-z^{-1}} + K_D \frac{2}{Ts} \frac{1-z^{-1}}{1+z^{-1}}$$

$$G(z) = \frac{U(z)}{E(z)}$$

$$u(k) = u_P(k) + u_I(k) + u_D(k)$$

$$u(k) = u_P(k) + u_I(k) + u_D(k) = e(k) \times G(z)$$

5. Phụ lục

- Khâu P:

$$u_P(k) = K_P e(k)$$

5. Phụ lục

- Khâu I:

$$u_I(k) = K_I \frac{T_s}{2} \frac{1+z^{-1}}{1-z^{-1}} e(k)$$

$$\Leftrightarrow u_I(k)(1-z^{-1}) = K_I \frac{T_s}{2}(1+z^{-1})e(k)$$

$$\Leftrightarrow u_I(k) - u_I(k-1) = K_I \frac{T_s}{2}\left[e(k)+e(k-1)\right]$$

$$\Leftrightarrow u_I(k) = K_I \frac{T_s}{2}\left[e(k)+e(k-1)\right] + u_I(k-1)$$

5. Phụ lục

- Khâu D:

$$u_D(k) = K_D \frac{2}{Ts} \frac{1-z^{-1}}{1+z^{-1}} e(k)$$

$$\Leftrightarrow u_D(k)(1+z^{-1}) = K_D \frac{2}{Ts}(1-z^{-1})e(k)$$

$$\Leftrightarrow u_D(k) + u_D(k-1) = K_D \frac{2}{Ts}[e(k) - e(k-1)]$$

$$\Leftrightarrow u_D(k) = K_D \frac{2}{Ts}[e(k) - e(k-1)] - u_D(k-1)$$

5. Phụ lục

- Tài liệu tham khảo bộ điều khiển PID số:

1. First Link
2. Second Link

- Trong bài viết đầu tiên, họ sử dụng phương pháp xấp xỉ Euler để triển khai bộ điều khiển PID số và họ không tách ra từng khâu dẫn đến khi chuyển từ hàm truyền rời rạc sang công thức sai phân phức tạp hơn

5. Phụ lục

- Trong cả 2 tài liệu họ đều thêm bộ lọc thông thấp vào khâu D và trong tài liệu thứ 2 còn thêm vào bộ giảm vọt lố gây ra do khâu I

5. Phụ lục

- Bộ ngăn vọt lố do khâu I

- Các bạn xem video sau để hiểu về lý do cần thiết kế bộ này: Anti-Windup

5. Phụ lục

- Bộ lọc thông thấp cho khâu D

- Các bạn xem video sau để hiểu về lý do cần thiết kế bộ này: LowPassFilter

5. Phụ lục

• Bộ lọc thông thấp cho khâu D

- Tài liệu tham khảo về thiết kế bộ lọc thông thấp số: LowPassFilter

- Trong tài liệu, họ đã trình bày về bộ lọc thông thấp và ảnh hưởng của nó lên hệ thống (chậm pha) các bạn có thể tìm hiểu thêm kiến thức ở chương 4 môn Tín hiệu và hệ thống để hiểu kỹ hơn. Ở đây ta đi vào cách triển khai bộ lọc thông thấp bậc nhất số

5. Phụ lục

• Bộ lọc thông thấp bậc nhất số:

- Bộ lọc thông thấp bậc nhất có dạng:

$$H(s) = \frac{\omega_c}{s + \omega_c} = \frac{Y(s)}{X(s)}$$

Với $\omega_c = 2\pi f_c$ là tần số cắt (rad/s)

y là tín hiệu ra (tín hiệu đã được lọc)

x là tín hiệu vào (tín hiệu chưa được lọc)

5. Phụ lục

- Xấp xỉ rời rạc theo quy tắc hình thang, ta được bộ lọc thông thấp rời rạc như sau:

Với:

$$s = \frac{2}{T_s}\frac{z-1}{z+1}$$

$$H(z) = \frac{\omega_c}{\frac{2}{T_s}\frac{z-1}{z+1} + \omega_c} = \frac{Y(z)}{X(z)}$$

5. Phụ lục

$$H(z) = \frac{\omega_c}{\frac{2}{T_s}\frac{z-1}{z+1} + \omega_c} = \frac{Y(z)}{X(z)}$$

$$\frac{Y(z)}{X(z)} = \frac{\omega_c}{\frac{1}{T_s}\frac{2(z-1)+\omega_c \times T_s \times (z+1)}{z+1}} = \frac{\omega_c \times T_s \times (z+1)}{2(z-1)+\omega_c \times T_s \times (z+1)}$$

$$\frac{Y(z)}{X(z)} = \frac{\omega_c \times T_s \times (1+z^{-1})}{2(1+z^{-1})+\omega_c \times T_s \times (1+z^{-1})}$$

5. Phụ lục

$$\frac{Y(z)}{X(z)} = \frac{\omega_c \times T_s \times (1+z^{-1})}{2(1+z^{-1}) + \omega_c \times T_s \times (1+z^{-1})}$$

$$= \frac{\omega_c T_s + \omega_c T_s z^{-1}}{(2+\omega_c T_s) - (2-\omega_c T_s) z^{-1}}$$

$$= \frac{\frac{\omega_c T_s}{2+\omega_c T_s} + \frac{\omega_c T_s}{2+\omega_c T_s} z^{-1}}{1 - \frac{2-\omega_c T_s}{2+\omega_c T_s} z^{-1}}$$

5. Phụ lục

- Phương trình sai phân của bộ lọc thông thấp:

$$\frac{y(k)}{x(k)} = \frac{\frac{\omega_c T_s}{2+\omega_c T_s} + \frac{\omega_c T_s}{2+\omega_c T_s} z^{-1}}{1 - \frac{2-\omega_c T_s}{2+\omega_c T_s} z^{-1}}$$

$$\Rightarrow y(k) - \frac{2-\omega_c T_s}{2+\omega_c T_s} y(k-1) = \frac{\omega_c T_s}{2+\omega_c T_s} x(k) + \frac{\omega_c T_s}{2+\omega_c T_s} x(k-1)$$

$$\Leftrightarrow y(k) = \frac{\omega_c T_s}{2+\omega_c T_s} x(k) + \frac{\omega_c T_s}{2+\omega_c T_s} x(k-1) + \frac{2-\omega_c T_s}{2+\omega_c T_s} y(k-1)$$

5. Phụ lục

- Triển khai bộ lọc và bộ điều khiển số theo dạng chính tắc:

- Ở phần trên, ta đã đi qua cách triển khai bộ điều khiển PID theo dạng trực tiếp. Tuy nhiên, cách triển khai trên có một vấn đề. Lấy bộ lọc thông thấp đã triển khai ở trên làm ví dụ

5. Phụ lục

- Dạng trực tiếp

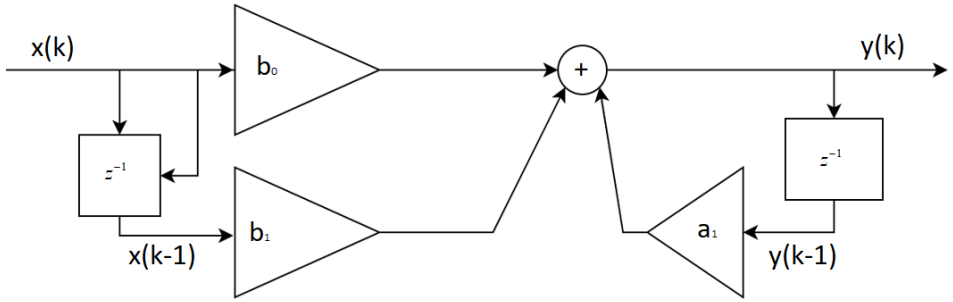
$$y(k) = \frac{\omega_c T_s}{2+\omega_c T_s} x(k) + \frac{\omega_c T_s}{2+\omega_c T_s} x(k-1) + \frac{2-\omega_c T_s}{2+\omega_c T_s} y(k-1)$$

$$\Leftrightarrow y(k) = b_0 x(k) + b_1 x(k-1) + a_1 y(k-1)$$

Với b0, b1 và a1 là các hằng số

5. Phụ lục

- Dạng trực tiếp



- Với mỗi mẫu x(k)

B1: Tính toán y(k)

B2: Lưu lại x(k-1) = x(k), y(k-1) = y(k)

5. Phụ lục

- Dạng chính tắc

$$H(z) = N(z)\frac{1}{D(z)} = \frac{Y(z)}{X(z)}$$

$$\Leftrightarrow \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} = \frac{Y(z)}{X(z)}$$

$$N(z) = b_0 + b_1 z^{-1}$$

$$D(z) = 1 - a_1 z^{-1}$$

- Ta có W(z) là tín hiệu trung gian

Với quan hệ: W(z)D(z) = X(z) và Y(z) = W(z)N(z)

5. Phụ lục

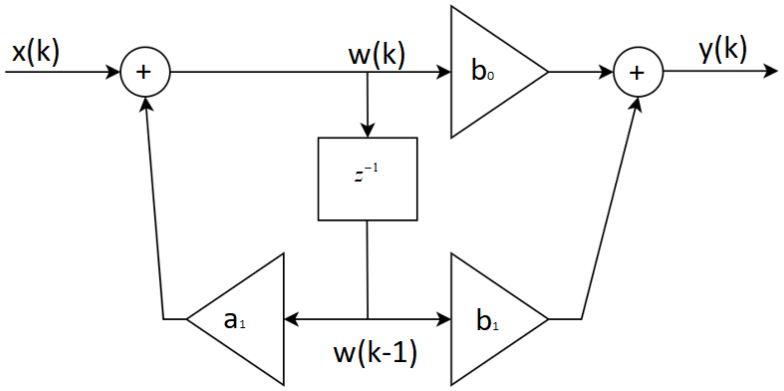
• Dạng chính tác:

- Phương trình sai phân:

$$w(k) = x(k) + a_1 w(k-1)$$
$$y(k) = b_0 w(k) + b_1 w(k-1)$$

5. Phụ lục

- Dạng chính tác:



- Với mỗi mẫu x(k):
- Bước 1: Tính toán w(k)
- Bước 2: Tính toán y(k)
- Bước 3: Lưu lại w(k-1) = w(k)

5. Phụ lục

- Nhận xét:

- Với dạng trực tiếp ta phải lưu lại 2 biến

x(k-1) và y(k-1) mỗi lần lấy 1 mẫu x(k) còn với dạng chính tắc với mỗi biến x(k) chỉ cần lưu lại 1 biến w(k-1). Tuy nhiên dạng chính tắc lại yêu cầu tính toán thêm 1 bước so với dạng trực tiếp, nhưng chỉ là phép toán đơn giản không quá ảnh hưởng đến tốc độ.

- Vậy với bộ lọc thông thấp bậc cao hơn thì sẽ cần phải lưu nhiều biến hơn (VD: x(k-1), x(k-2), y(k-1), y(k-2),v.v..) thì sử dụng dạng chính tắc sẽ phải lưu ít biến hơn mỗi lần lấy mẫu x(k)

5. Phụ lục

- Để tìm hiểu kỹ hơn các bạn tham khảo chương 6 môn Xử lý số tín hiệu