*Đề Tài Ứng Dụng*

# TEMPERATURE CONTROLLER

_Hoàng Anh Hiệp_

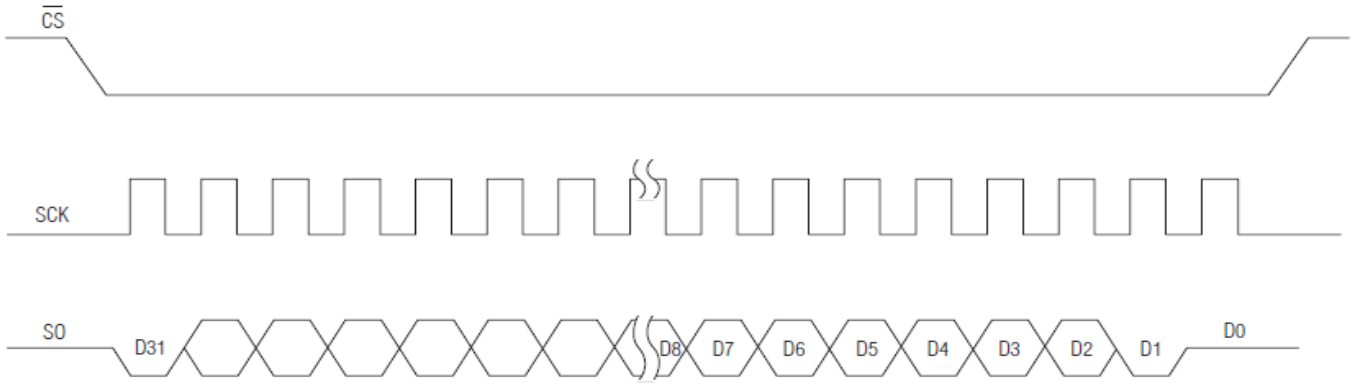
# 1. Thiết bị phần cứng:

- Development Kit STM32F4 Discovery
- IC Transducer  MAX31855
- SSR-40 DA
- Thermocoup
- Heater
  - Sơ đồ mạch nguyên lý:

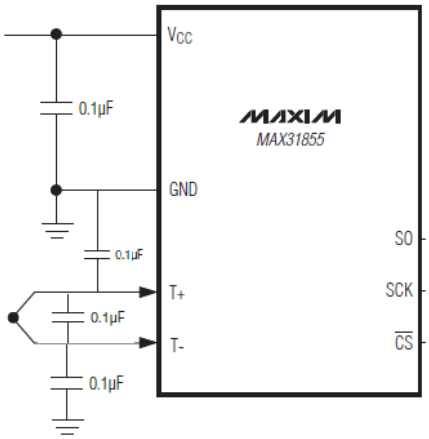
- Max318 cho ra tín hiệu nhiệt độ qua giao tiếp SPI đã chỉnh sửa. Bảng phân phối một gói dữ liệu của Max318 như sau:

| | 14-BIT THERMOCOUPLE TEMPERATURE DATA | | | | RES | FAULT BIT | 12-BIT INTERNAL TEMPERATURE DATA | | | | RES | SCV BIT | SCG BIT | OC BIT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **BIT** | D31 | D30 | ... | D18 | D17 | D16 | D15 | D14 | ... | D4 | D3 | D2 | D1 | D0 |
| **VALUE** | Sign | MSB $2^{10}$ (1024°C) | ... | LSB $2^{-2}$ (0.25°C) | Reserved | 1 = Fault | Sign | MSB $2^{6}$ (64°C) | ... | LSB $2^{-4}$ (0.0625°C) | Reserved | 1 = Short to $V_{CC}$ | 1 = Short to GND | 1 = Open Circuit |

- Gói dữ liệu của Max318 đưa ra là 32 bit nhưng thanh ghi SPI của STM32F4 lại chỉ có 16 bit, vì vậy sau nhiều lần thử nghiệm đọc bằng SPI không thành công nên đã chuyển sang đọc dữ liệu bằng cách tự kích xung và tự đọc tín hiệu vào từ MISO theo sơ đồ sau.




- Trong quá trình thực nghiệm đọc dữ liệu thu được từ Max318 nhận thấy tín hiệu nhiệt độ từ thermocoup không ổn định, thường dao động khoảng $2^0$ C và tín hiệu nhiệt độ dường như bị offset 1 khoảng so với nhiệt độ thật. Nhận thấy tình trạng trên chỉ xảy ra khi sử dụng nguồn lưới hoặc khi gắn thermocoup vào đối tượng bằng sắt, nên đã giải quyết bằng phương pháp thêm 1 tụ 104 giữa 2 chân T- và T+ để loại bỏ nhiễu do thermocoup tiếp xúc đối tượng bằng sắt, đồng thời thêm 2 tụ 104 vào mỗi chân T+ và T- nối mass giải quyết vấn đề nhiễu do nguồn lưới.



## 2. Thiết kế phần mềm điều khiển:

### a) Các hàm đã viết trên Keil uVision 5 sử dụng thư viện STM32 Peripheral Driver:

- File IO_init.c:
  - Hàm Temp_SPI_init() tạo liên kết GPIO cho các chân để đọc từ Max318.
  - Hàm PWM_Init() tạo liên kết GPIO cho chân PA8 để xuất xung PWM đồng thời setting Timer 1 cho chức năng PWM.
  - Hàm Sample_Timer_init() tạo thời gian lấy mẫu 2s bằng Timer 6.
  - Hàm user_bt_init() tạo liên kết GPIO cho user button nối với PA0 và tạo interupt cho PA0.
- File IT.c:
  - Hàm TIM6_DAC_IRQHandler() là interupt cho timer 6, chỉ thực hiện việc xóa cờ interupt và đánh dấu cờ Timer6_IT do mình điều khiển.
  - Hàm EXTI0_IRQHandler() là interupt cho user button để đánh dấu cờ tunning cho auto tunning.
- File Function.c:
  - Hàm GetTemp() xuất xung và đọc về giá trị nhiệt độ từ max318.
  - Hàm AverTemp() lấy 30 mẫu nhiệt độ và tính ra giá trị trung bình AvrAmbTemp (Nhiệt độ IC) và AvrThermoTemp (Nhiệt độ Thermocoup).
  - Hàm SampleTime_1s() chu kì lấy mẫu 2s
  - Hàm PID_controller() bộ điều khiển PID.
  - Hàm PID_Tuner() bộ auto tuner cho PID.
  - Hàm PID_Fuzzy_SelfTuning() bộ fuzzy self tuning cho PID.
  - Hàm PD_Fuzzy() bộ fuzzy PD sử dụng Hybrid với PID fuzzy self tuning.
  - Hàm Read_Flash() và Write_Flash() để đọc và ghi vào flash.

### b) Thiết kế bộ điều khiển PID:

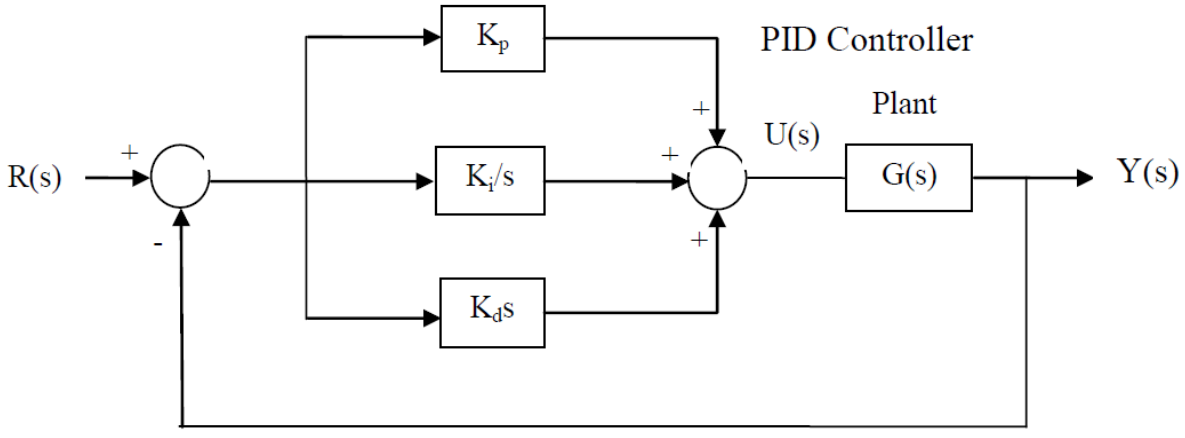
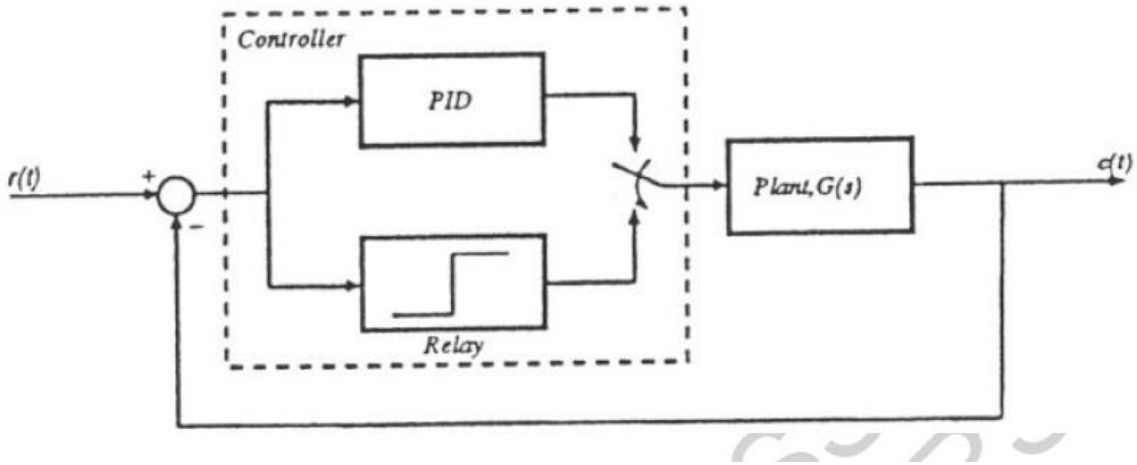



Fig. (1) Closed loop system with PID

- Bộ điều khiển PID khá đơn giản, ứng dụng chạy được ngay, nhưng lại rất khó tìm được điểm làm việc tối ưu.
- Khi tìm được điểm làm việc tương đối ổn định thì khả năng chống chịu khi có tải khá kém, khi hoạt động thật trên máy đóng gói ở 120$^o$ C nhiệt độ giảm > 10$^o$ C và rất chậm tăng lên lại, đồng thời vọt lố lúc khởi động có thể lên đến > 20$^o$ C

⇨ Bộ điều khiển PID thiết kế đơn giản nhưng vân hành không hiệu quả, không thích ứng được với môi trường thay đổi, và phải tự chỉnh mỗi lần thay đổi đối tượng sử dụng.

**c) Thiết kế bộ điều khiển PID auto tuning bằng phương pháp relay:**




- Phương pháp thiết kế.
- Phương pháp auto tuning bằng relay cho thông số PID chạy được ổn định, có thể tuning để chạy được với các đối tượng khác nhau, nhưng vẫn chứa nhược điểm của bộ PID thông thường như khả năng chịu tải kém.

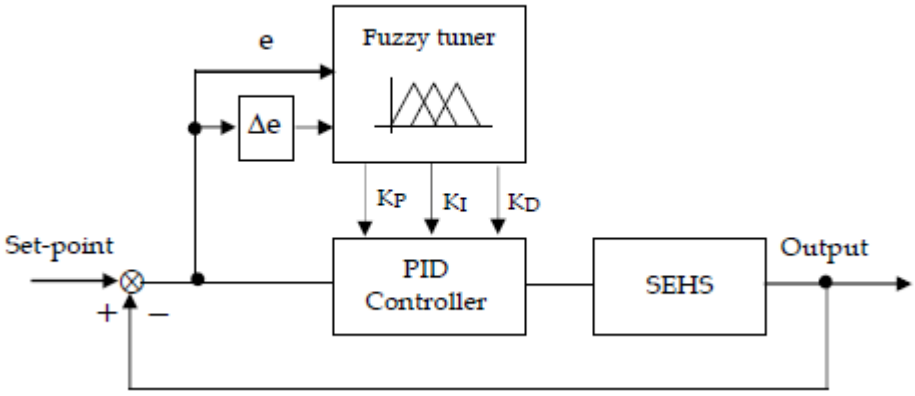
⇨ Tuy đã giải quyết được vấn đề về đối tượng nhưng thông số của bộ tuner đưa ra chưa tối ưu, như thời gian lên khá chậm, vọt lố khá cao, dễ mất ổn định khi tải lớn.

d) **Thiết kế bộ điều khiển PID auto tuning bằng phương pháp Nelder Mead (Chưa thực hiện):**

- Phương pháp thực hiện.
- Phương pháp Nelder Mead còn gọi là phương pháp lên đỉnh hay xuống dốc, vì theo thời gian tuning nó sẽ cho ra thông số PID tối ưu nhất phù hợp với yêu cầu của người vận hành.

⇨ Tuy phương pháp này có vẻ như tốt hơn auto tuning bằng relay nhưng do bản chất vẫn chỉ là auto tuning nên chắc chắn khả năng chống chịu thay đổi lúc hoạt động vẫn sẽ kém nên không tập trung vào những nghiên cứu này nữa.

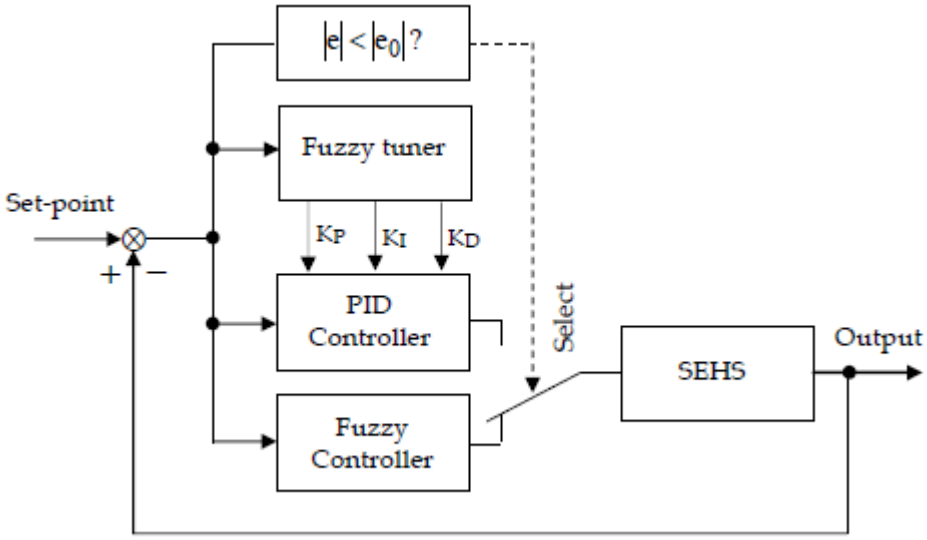
e) **Thiết kế bộ điều khiển PID Fuzzy self tuning( Đã chọn cho hoạt động) :**




- Phương pháp thiết kế
- Phương pháp này sử dụng 2 bộ điều khiển chạy song song nhau, bộ điều khiển PD Fuzzy sẽ lấy thông số nhiệt độ và độ thay đổi nhiệt độ để cho ra thông số Kp, Ki, Kd tương ứng, từ đó bộ điều khiển PID sẽ tiếp tục điều khiển đối tượng.
- Do đặc điểm tùy chỉnh các thông số PID nên bộ điều khiển này có khả năng thích ứng hoàn hảo, cả khả năng đáp ứng thời gian lẫn khả năng chịu tải, nhiễu từ môi trường đều rất tốt, nhưng nó vẫn còn 1 khuyết điểm là các luật fuzzy điều khiển tùy chỉnh bằng kinh nghiệm của người thiết kế, bằng phương pháp thử sai là chủ yếu nên các luật được thiết kế cho bộ điều khiển chưa phải là tối ưu.

  ⇨ Phương pháp này cho khả năng điều khiển tối ưu nhất nhưng nên kết hợp với 1 phương pháp tối ưu hóa các bộ luật cho fuzzy như Nelder Mead chẳng hạn, có thể sẽ cho ra bộ điều khiện tự động hoàn toàn và có thông số hạt động chắn chắn nhất, đáng tin cậy được.

f) **Thiết kế bộ điều khiển Hybrid PID Fuzzy self tuning & Fuzzy ( Thất bại) :**




- Phương pháp thiết kế

- Do khó khăn trong việc tối ưu hóa cho bộ PID fuzzy self tuning nên đã tìm ra 1 giải pháp khác là cho chạy Hybrid với bộ Fuzzy thông thường. Bộ Fuzzy PD thường sẽ chạy trong miền nhiệt độ < 0.7 *( nhiệt độ đặt), và phần còn lại sẽ do bộ PID fuzzy self tuning đảm nhiệm. Vì thiết kế vậy nên sẽ giảm áp lực lên bộ điều khiển PID fuzzy self tuning.
- Thực nghiệm cho thấy việc thêm bộ điều khiển Fuzzy PD vào lại làm giảm tính linh hoạt của bộ điều khiển vì luật điều khiển của nó lại phải phụ thuộc vào đối tượng, vấn để lớn hơn là sự mất ổn định của bộ PID fuzzy self tuning khi chuyển từ fuzzy PD sang.

⇨ Mặc dù sự kết hợp giữa 2 mô hình này không cho kết quả tốt nhưng việc chạy Hybrid các giải thuật điều khiển khác nhau đang là xu thế vì nó dễ thực hiện hơn và cho đáp ứng tốt hơn các bộ điều khiển đòi hỏi phương trình toán học phức tạp để thiết kế.

**g) Thiết kế bộ điều khiển Hybrid <u>PID Fuzzy self tuning</u> dùng tham số <u>auto tuning từ Nelder Mead</u> để tự chỉnh tỷ lệ cho các biến vào Fuzzy (chưa thực hiện)**

- Đầu tiên cho hệ thống chạy ở chế độ auto tuning, sau khi đã đạt được thông số PID tối ưu, ta sẽ cho chạy tiếp PID fuzzy self tuning nhưng với thông số PID nằm trong miền gần với thông số tối ưu đã có được từ auto tuning.

# 3. <u>Kết luận:</u>

- Chương trình đang chạy vẫn đang sử dụng bộ PID Fuzzy self tuning với các biến mờ khá đơn giản và sử dụng giải mờ bằng phương pháp trung bình có trọng số. Bộ PID Fuzzy self tuning chưa phải là tối ưu nên có thể tiếp tục cải tiến bộ luật và nếu có thể thì thay phương pháp giải mờ thành centroid sẽ cho đáp ứng tốt hơn hiện tại.
- Chương trình vẫn còn giữ chức năng auto tuning bằng relay, mỗi lúc xóa hết toàn bộ flash thì nó sẽ tự tuning lại hoặc nhấn user button. Thông số tuning được vẫn chưa sử dụng cho bộ PID Fuzzy self tuning vì nó vẫn chưa phải là tối ưu.