BÁO CÁO THỰC HÀNH BÀI Lab1\_GPIO\_Timer

Môn học: **Chuyên đề thiết kế hệ thống nhúng 1** - Mã lớp: **CE437.P11**

Giảng viên hướng dẫn thực hành: **Phạm Minh Quân**

|  |  |  |
| --- | --- | --- |
| **Thông tin sinh viên** | **Mã số sinh viên** | **Họ và tên** |
| **22521472** | **Phạm Quốc Tiến** |
| **22521570** | **Trịnh Thành Trung** |
| **22521564** | **Nguyễn Đức Trung** |
| **Link các tài liệu tham khảo** *(nếu có)* |  | |
| **Đánh giá của giảng viên**:  *+ Nhận xét*  *+ Các lỗi trong chương trình*  *+ Gợi ý* |  | |

*[Báo cáo chi tiết các thao tác, quy trình sinh viên đã thực hiện trong quá trình làm bài thực hành. Chụp lại hình ảnh màn hình hoặc hình ảnh kết quả chạy trên sản phẩm. Mô tả và giải thích chương trình tương ứng để cho ra kết quả như hình ảnh đã trình bày.]*

Mục lục

[1) Giao tiếp UART 3](#_Toc181908909)

[2) Đọc tín hiệu ADC 3](#_Toc181908910)

[3) Giao tiếp I2C 3](#_Toc181908911)

[4) Đếm xung Encoder 5](#_Toc181908912)

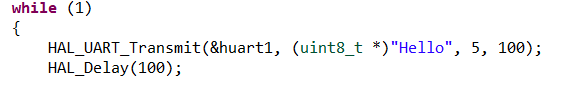
[5) Tạo xung PWM 6](#_Toc181908913)

[6) Bài tập 7](#_Toc181908914)

[7) Kết quả thực tế 8](#_Toc181908915)

1. Giao tiếp UART

Chương trình mà nhóm viết

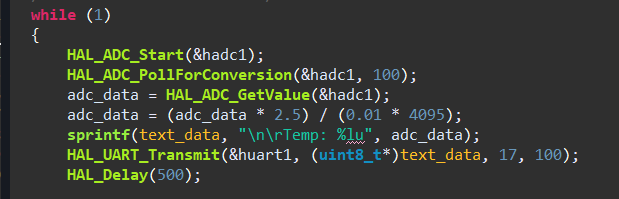


1. Đọc tín hiệu ADC

Khai báo



Chương trình trong hàm main

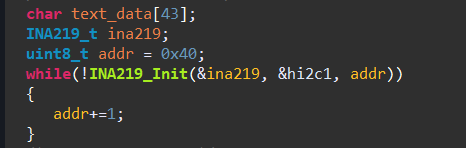


Do theo datasheet thì điện áp ADC đọc được thì sẽ chuyển sang giá trị nhiệt độ với tỉ lệ

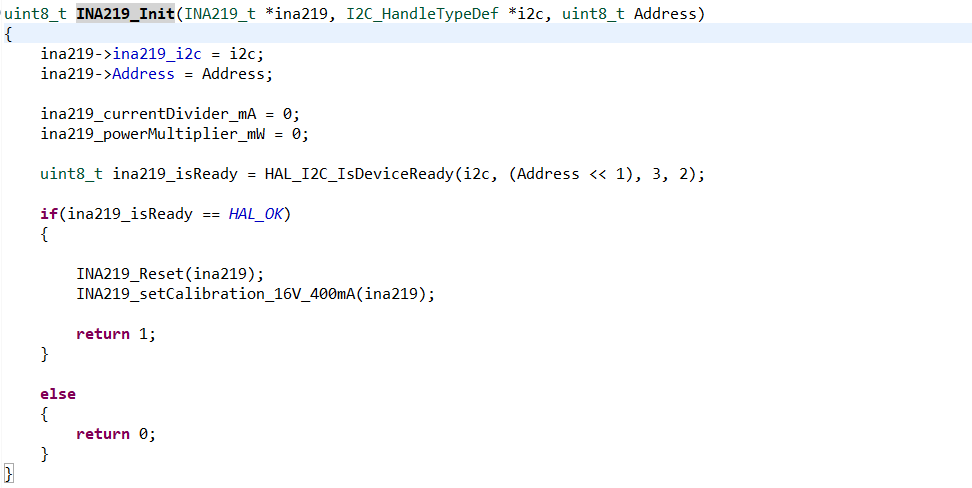
1°C = 10mV. Để tính được điện áp qua giá trị ADC thu được thì nhóm thực hiện chia cho giá trị lớn nhất – 4095 và nhân kết quả đó cho 2.5V (do trong datasheet mắc điện trở tương đương 0.5Ω cho LM35DZ).

1. Giao tiếp I2C

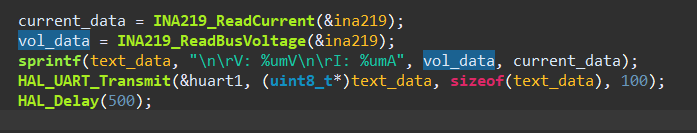
Nhóm sẽ sử dụng thư viện bên ngoài INA219.h. Đầu tiên nhóm sẽ khai báo INA219, do địa chỉ của INA219 sẽ có xe là địa chỉ mặc định (0x40) có xe là địa chỉ khác hoặc tệ hơn lả INA219 không giao tiếp được thì nhóm sẽ sử dụng vòng while để chặn việc gửi, nhận dữ liệu khi chưa giao tiếp được



Trong code khai báo INA219 của thư viện, chương trình gán handle hi2c1 vào ina219 và kiểm tra xem liệu INA219 có đang busy không. Nếu có chương trình sẽ gửi lệnh reset cho INA219 và sẽ set lại calibration với bus voltage range = 16V, PGA = 40mV, BADC = 12bit, SADC = 12bit 532μs và mode ‘shunt and bus, continuous’ để cảm biến có thể đo giá trị và chương trình của nhóm sẽ xử lý đúng data nhận được



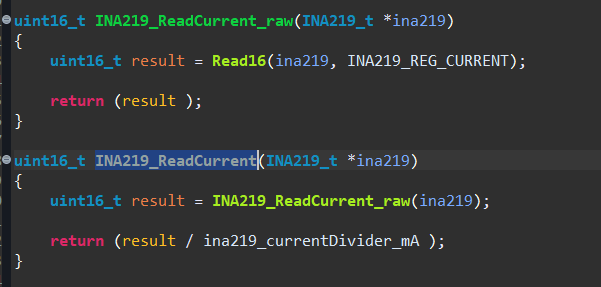
Tiếp trong hàm while nhóm sẽ đọc dòng và điện áp, sau đó xuất lên màn oled



Hàm ReadCurrent sẽ convert raw data từ ReadCurrent\_Raw bằng cách chia cho

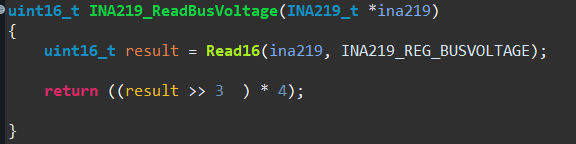
ina219\_currentDivider\_mA = 20, do data đọc được sẽ là 50µA mỗi bit mà chuyển sang mA

thì sẽ cần lấy



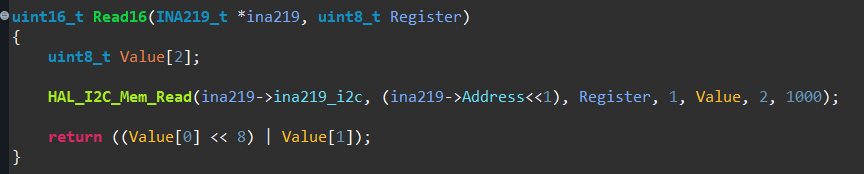
Hàm ReadBusVoltage sẽ đọc áp và chuyển về mV, lý do phải dịch phải 3 bit rồi mới nhân 4 vì

vì 8bit trọng số thấp của thanh ghi Bus không dùng để lưu giá trị điện áp mà dành cho mục đích khác và việc nhân với 4 do mỗi đơn vị trong kết quả đo bằng 4mV



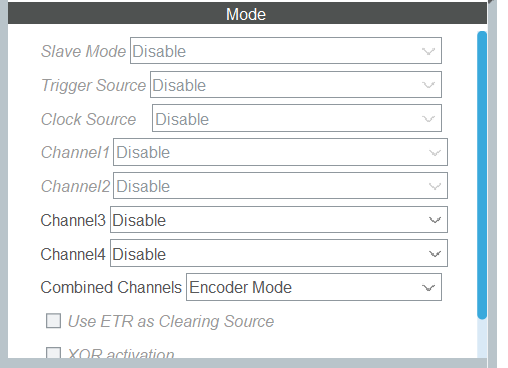
Hàm Read16 chỉ đơn giản là đọc giá trị ở địa chỉ thanh ghi, phần địa chỉ dịch trái để thành 8bit

và (Value[0] << 8) | Value[1]) để ghép thành số uint16\_t

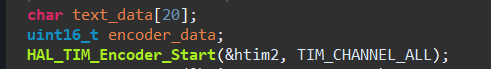


1. Đếm xung Encoder

Nhóm sẻ setup cho Timer2 hoạt động ở chế độ encoder



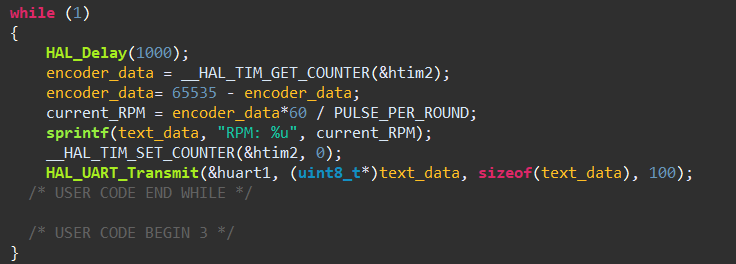
Khai báo trong hàm main



Trong hàm while, nhóm sẽ tiến hành đọc và in ra màn hình mỗi 1s.

Đầu tiên nhóm sẽ tiến hành đọc giá trị thanh ghi counter của timer2, do khi code test encoder thì nhóm thấy encoder của xe sẽ đếm ngược lại khi tiến lên, vậy nên khi lấy data encoder thì nhóm sẽ trừ cho 65535 để lấy được giá trị thực.

Để tính round per minute thì nhóm sẽ tiến hành lấy data encoder \* 60 / PULSE\_PER\_ROUND (số xung trả về khi bánh xe quay được 1 vòng).

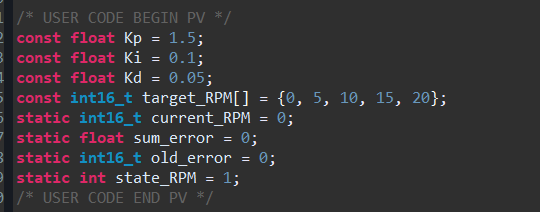


1. Tạo xung PWM

Ở phần này nhóm sẽ sử dụng thêm 1 timer chế độ interrupt để thay đổi tốc độ động cơ mỗi 2s.

Theo datasheet thì khi set giá trị khác 0 cho R\_PWM thì xe sẽ tiến lên, L\_PWM thì ngược lại, vậy nên nhóm sẽ chỉ làm việc với channel\_4 của timer2.

Khai báo

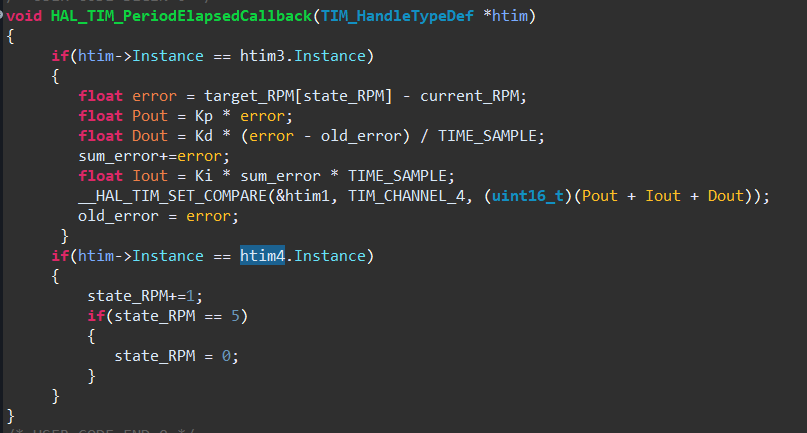


Code trong hàm ngắt

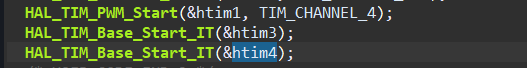
Ở đây nhóm sẻ thực hiện điều chỉnh giá trị pwm thông qua bộ điều khiển PID. Do tích phân không có trong code c, vậy nên nhóm sẽ sử dụng phép tính tương tương đó là lấy

TIME\_SAMPLE sẽ bằng 100ms do PID sẽ chạy sau mỗi 100ms.

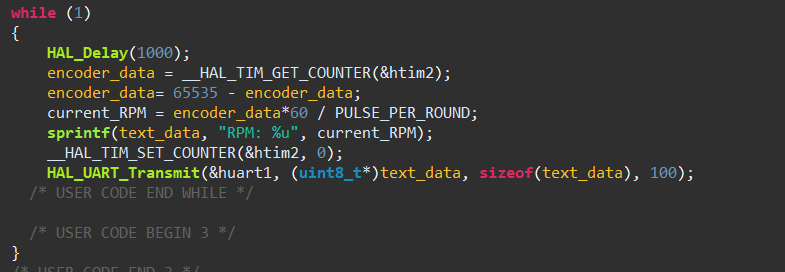
Và sau mỗi 2s thì sẽ set state\_RPM thành giá trị khác



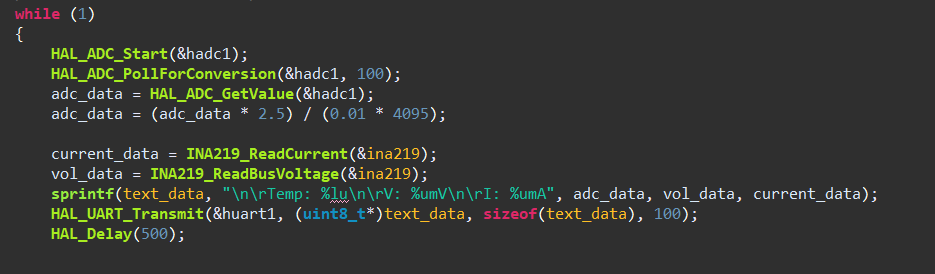
Bật timer trong hàm main



Để biết giá trị RPM thì nhóm sẽ tái sử dụng code của phần 5



1. Bài tập

[Sensor node](https://github.com/kamuisi/CDHTN/tree/main/lab_2/sensor_node): phần này nhóm sẽ sử dụng lại hai phần code sensor và code uart trước đó với một số thay đổi  


[Actuator node](https://github.com/kamuisi/CDHTN/tree/main/lab_2/actuator_node): phần này nhóm sẽ tái sử dụng hai phần code actuator và code uart

1. Kết quả thực tế

[Sensor node](https://drive.google.com/file/d/1mh0VG_iXwZqthiT8wDtEU-j4uMnAbFhl/view?usp=sharing)

[Actuator node](https://drive.google.com/file/d/1mX_wt5Z3KHCdnF2FRiPEmY3HiJn6Y0XK/view?usp=sharing)