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

Chương trình mà nhóm viết

```
while (1)
{
    HAL_UART_Transmit(&huart1, (uint8_t *)"Hello", 5, 100);
    HAL_Delay(100);
```

2) Đọc tín hiệu ADC

Khai báo

```
/* USER CODE BEGIN PV */
static uint32_t adc_data = 0;
```

Chương trình trong hàm main

```
while (1)
{
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, 100);
    adc_data = HAL_ADC_GetValue(&hadc1);
    adc_data = (adc_data * 2.5) / (0.01 * 4095);
    sprintf(text_data, "\n\rTemp: %lu", adc_data);
    HAL_UART_Transmit(&huart1, (uint8_t*)text_data, 17, 100);
    HAL_Delay(500);
```

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).

3) 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

```
char text_data[43];
INA219_t ina219;
uint8_t addr = 0x40;
while(!INA219_Init(&ina219, &hi2c1, addr))
{
   addr+=1;
}
```

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

```
uint8_t INA219_Init(INA219_t *ina219, I2C_HandleTypeDef *i2c, uint8_t Address)
{
    ina219->ina219_i2c = i2c;
    ina219->Address = Address;

    ina219_currentDivider_mA = 0;
    ina219_powerMultiplier_mW = 0;

    uint8_t ina219_isReady = HAL_I2C_IsDeviceReady(i2c, (Address << 1), 3, 2);

    if(ina219_isReady == HAL_OK)
    {

        INA219_Reset(ina219);
        INA219_setCalibration_16V_400mA(ina219);

        return 1;
    }

    else
    {
        return 0;
    }
}
```

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

```
current_data = INA219_ReadCurrent(&ina219);
vol_data = INA219_ReadBusVoltage(&ina219);
sprintf(text_data, "\n\rV: %umV\n\rI: %umA", vol_data, current_data);
HAL_UART_Transmit(&huart1, (uint8_t*)text_data, sizeof(text_data), 100);
HAL_Delay(500);
```

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 $\frac{1000\mu A}{50\mu A} = 20$

```
uint16_t INA219_ReadCurrent_raw(INA219_t *ina219)
{
    uint16_t result = Read16(ina219, INA219_REG_CURRENT);

    return (result );
}

uint16_t INA219_ReadCurrent(INA219_t *ina219)
{
    uint16_t result = INA219_ReadCurrent_raw(ina219);

    return (result / ina219_currentDivider_mA );
}
```

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

```
uint16_t INA219_ReadBusVoltage(INA219_t *ina219)
{
    uint16_t result = Read16(ina219, INA219_REG_BUSVOLTAGE);

    return ((result >> 3  ) * 4);

}
```

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

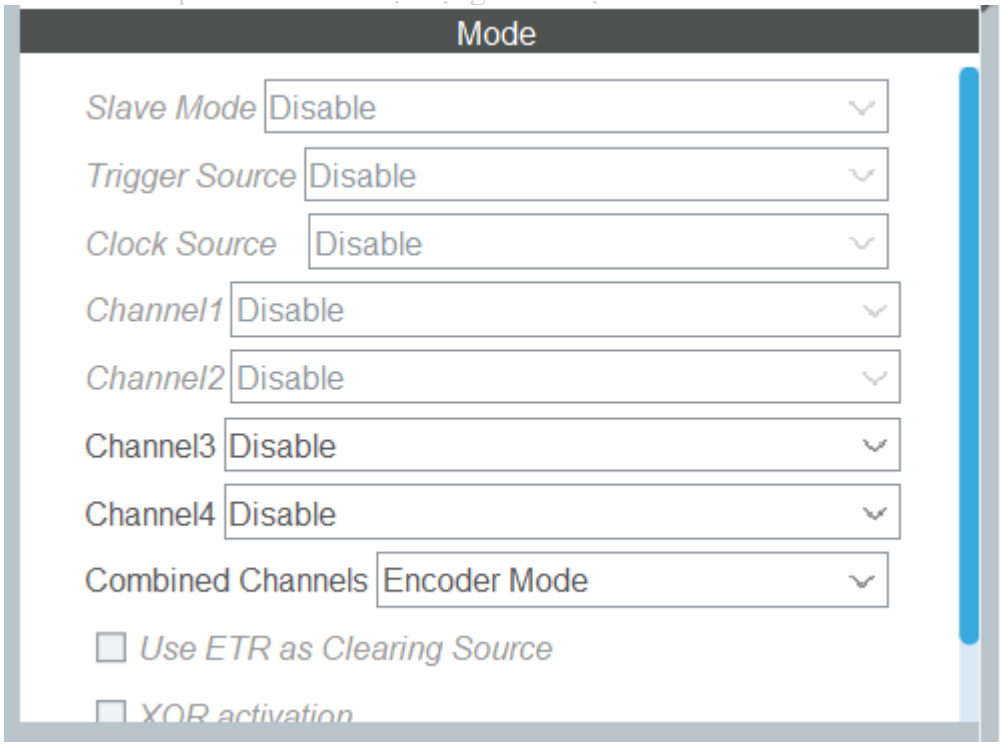
```
uint16_t Read16(INA219_t *ina219, uint8_t Register)
{
    uint8_t Value[2];

    HAL_I2C_Mem_Read(ina219->ina219_i2c, (ina219->Address<<1), Register, 1, Value, 2, 1000);

    return ((Value[0] << 8) | Value[1]);
}
```

4) Đếm xung Encoder

Nhóm sẻ setup cho Timer2 hoạt động ở chế độ encoder



Khai báo trong hàm main

```
char text_data[20];
uint16_t encoder_data;
HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_ALL);
```

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).

```
while (1)
{
    HAL_Delay(1000);
    encoder_data = __HAL_TIM_GET_COUNTER(&htim2);
    encoder_data= 65535 - encoder_data;
    current_RPM = encoder_data*60 / PULSE_PER_ROUND;
    sprintf(text_data, "RPM: %u", current_RPM);
    __HAL_TIM_SET_COUNTER(&htim2, 0);
    HAL_UART_Transmit(&huart1, (uint8_t*)text_data, sizeof(text_data), 100);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
```

5) 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

```
/* USER CODE BEGIN PV */
const float Kp = 1.5;
const float Ki = 0.1;
const float Kd = 0.05;
const int16_t target_RPM[] = {0, 5, 10, 15, 20};
static int16_t current_RPM = 0;
static float sum_error = 0;
static int16_t old_error = 0;
static int state_RPM = 1;
/* USER CODE END PV */
```

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

$$Ki * \text{tổng tích lũy của error} * \text{thời gian giữa các mẫu}$$

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

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
     if(htim->Instance == htim3.Instance)
     {
        float error = target_RPM[state_RPM] - current_RPM;
        float Pout = Kp * error;
        float Dout = Kd * (error - old_error) / TIME_SAMPLE;
        sum_error+=error;
        float Iout = Ki * sum_error * TIME_SAMPLE;
        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, (uint16_t)(Pout + Iout + Dout));
        old_error = error;
      }
     if(htim->Instance == htim4.Instance)
     {
         state_RPM+=1;
         if(state_RPM == 5)
         {
            state_RPM = 0;
         }
     }
}
```

Bật timer trong hàm main

```
  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
  HAL_TIM_Base_Start_IT(&htim3);
  HAL_TIM_Base_Start_IT(&htim4);
```

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

```
while (1)
{
    HAL_Delay(1000);
    encoder_data = __HAL_TIM_GET_COUNTER(&htim2);
    encoder_data= 65535 - encoder_data;
    current_RPM = encoder_data*60 / PULSE_PER_ROUND;
    sprintf(text_data, "RPM: %u", current_RPM);
    __HAL_TIM_SET_COUNTER(&htim2, 0);
    HAL_UART_Transmit(&huart1, (uint8_t*)text_data, sizeof(text_data), 100);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
```

6) Bài tập

Sensor node: phần này nhóm sẽ sử dụng lại hai phần code sensor và code uart trước đó với một số thay đổi

```
while (1)
{
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, 100);
    adc_data = HAL_ADC_GetValue(&hadc1);
    adc_data = (adc_data * 2.5) / (0.01 * 4095);

    current_data = INA219_ReadCurrent(&ina219);
    vol_data = INA219_ReadBusVoltage(&ina219);
    sprintf(text_data, "\n\rTemp: %lu\n\rV: %umV\n\rI: %umA", adc_data, vol_data, current_data);
    HAL_UART_Transmit(&huart1, (uint8_t*)text_data, sizeof(text_data), 100);
    HAL_Delay(500);
```

Actuator node: phần này nhóm sẽ tái sử dụng hai phần code actuator và code uart

7) Kết quả thực tế

Sensor node
Actuator node