**TỔNG QUAN CÁC BƯỚC**

Contents

[**Bước 1:** Khai báo các thư viện; định nghĩa macro để sử dụng **TIMER, ADC.** 2](#_Toc144678171)

[**1. Khai báo các thư viện** 2](#_Toc144678172)

[**2. Định nghĩa MACRO** 2](#_Toc144678173)

[**Bước 2:** Cấu hình để sử dụng ngoại vi **TIMER, ADC**, bộ lọc **Kalman.** 2](#_Toc144678174)

[**1. Cấu hình ADC1** 2](#_Toc144678175)

[**2. Cấu hình TIMER1** 2](#_Toc144678176)

[**3. Khởi tạo bộ lọc Kalman** 3](#_Toc144678177)

[**Bước 3:** Lấy giá trị cường độ ánh sáng của cảm biến ánh sáng. 3](#_Toc144678178)

[**Bước 4:** Xử lý dữ liệu của cường độ ánh sáng, điều khiển độ sáng của LED. 3](#_Toc144678179)

[**Bước 5:** Tạo chu kỳ thời gian 100 ms để điều khiển giá trị độ sáng của LED. 4](#_Toc144678180)

**THUYẾT MINH**

# **Bước 1:** Khai báo các thư viện; định nghĩa macro để sử dụng **TIMER, ADC.**

## **1. Khai báo các thư viện**

**-** Thư viện C : math.h - Sử dụng hàm toán học chuyển đổi giá trị CDDASS thành độ sáng LED.

**-** Thư viện hệ thống STM32: system\_stm32f4xx.h – Tạo xung clock hệ thống, timer.h – Xử lý sự kiện thời gian.

**-** Thư viện điều khiển các ngoại vi: stm32f401re\_rcc.h – Tạo xung clock cho các ngoại vi, stm32f401re\_tim.h – Cấu hình, sử dụng ngoại vi TIMER, stm32f401re\_adc.h – Cấu hình, sử dụng ngoại vi ADC, stm32f401re\_gpio.h – Cấu hình, sử dụng GPIO.

**-** Thư viện sử dụng bộ lọc Kalman: kalman\_filter.h

## **2. Định nghĩa MACRO**

**-** Định nghĩa các MACRO sử dụng TIM1\_CH4.

- Định nghĩa các MACRO sử dụng ADC1.

# **Bước 2:** Cấu hình để sử dụng ngoại vi **TIMER, ADC**, bộ lọc **Kalman.**

## **1. Cấu hình ADC1**

B1: Cấu hình chân GPIO\_Pin\_5 của Port C ở chế độ Analog.

B2: Cấu hình ADC1

+ Cấu hình độ phân giải 12 bits: ADC\_InitStructure.ADC\_Resolution = ADC\_Resolution\_12b;

+ Cấu hình chế độ quét đơn kênh: ADC\_InitStructure.ADC\_ScanConvMode = *DISABLE*;

+ Cấu hình chuyển đổi dữ liệu liên tục: ADC\_InitStructure.ADC\_ContinuousConvMode = *ENABLE*;

+ Cấu hình kênh ADC\_IN15 kết nối với cảm biến ánh sáng, thời gian lấy mẫu 15 cycles: ADC\_RegularChannelConfig(ADCx\_SENSOR, ADC\_Channel\_15, 1, ADC\_SampleTime\_15Cycles);

## **2. Cấu hình TIMER1**

B1: Cấu hình chân GPIO\_Pin\_11 của Port A là TIM1\_CH4.

B2: Cấu hình TIMER1 ở chế độ Basic với tần số hoạt động 10 KHz.

+ Cấu hình chế độ đếm lên: TIM\_TimeBaseInitStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;

+ Cấu hình hệ số chia tần: TIM\_TimeBaseInitStructure.TIM\_Prescaler = 0;

+ Cấu hình chu kỳ update events: TIM\_TimeBaseInitStructure.TIM\_Period = PERIOD;

B3: Cấu hình TIMER1 hoạt động ở chế độ PWM

+ Cấu hình TIMER hoạt động ở chế độ PWM2: TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM2;

+ Cấu hình độ rộng xung ban đầu bằng 0: TIM\_OCInitStructure.TIM\_Pulse = 0;

+ Cấu hình độ cực đầu ra: TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_Low;

## **3. Khởi tạo bộ lọc Kalman**

+ Chọn giá trị e\_mea – giá trị sai lệch so với giá trị chuẩn: 90 lux.

+ Chọn giá trị e\_est – giá trị chọn băng với e\_mea: 90 lux.

+ Chọn giá trị q – tốc độ biến thiên của đối tượng đo: 0.2.

KalmanFilterInit(90, 90, 0.2);

# **Bước 3:** Lấy giá trị cường độ ánh sáng của cảm biến ánh sáng.

\* @func LightSensor\_AdcPollingRead

\* @brief To read the value of light sensor

\* @param None

\* @retval Light value

**static** uint32\_t **LightSensor\_AdcPollingRead**(**void**)

**\* Thuật toán:**

B1: Bắt đầu quá trình chuyển đổi dữ liệu ADC.

B2: Đợi quá trình chuyển đổi dữ liệu hoàn tất để đọc dữ liệu.

B3: Đọc giá trị cường độ ánh sáng chuyển đổi.

# **Bước 4:** Xử lý dữ liệu của cường độ ánh sáng, điều khiển độ sáng của LED.

\* @func ABL\_Process

\* @brief To read, process the value of ADC and adjust the light value of LED

\* @param None

\* @retval None

**static** **void** **ABL\_Process**(**void**)

**Thuật toán**

B1: Lấy dữ liệu cường độ ánh sáng đã chuyển đổi.

ambient\_light\_value = LightSensor\_AdcPollingRead();

B2: Đưa giá trị cường độ ánh sáng đó qua bộ lọc Kalman để lọc nhiễu.

+ Hàm bộ lọc Kalman dùng để lọc nhiễu

\* @func KalmanFilter

\* @brief To filter the noise of brightness

\* @param Light\_value - The sensor value before filtering

\* @retval The sensor value after filtering

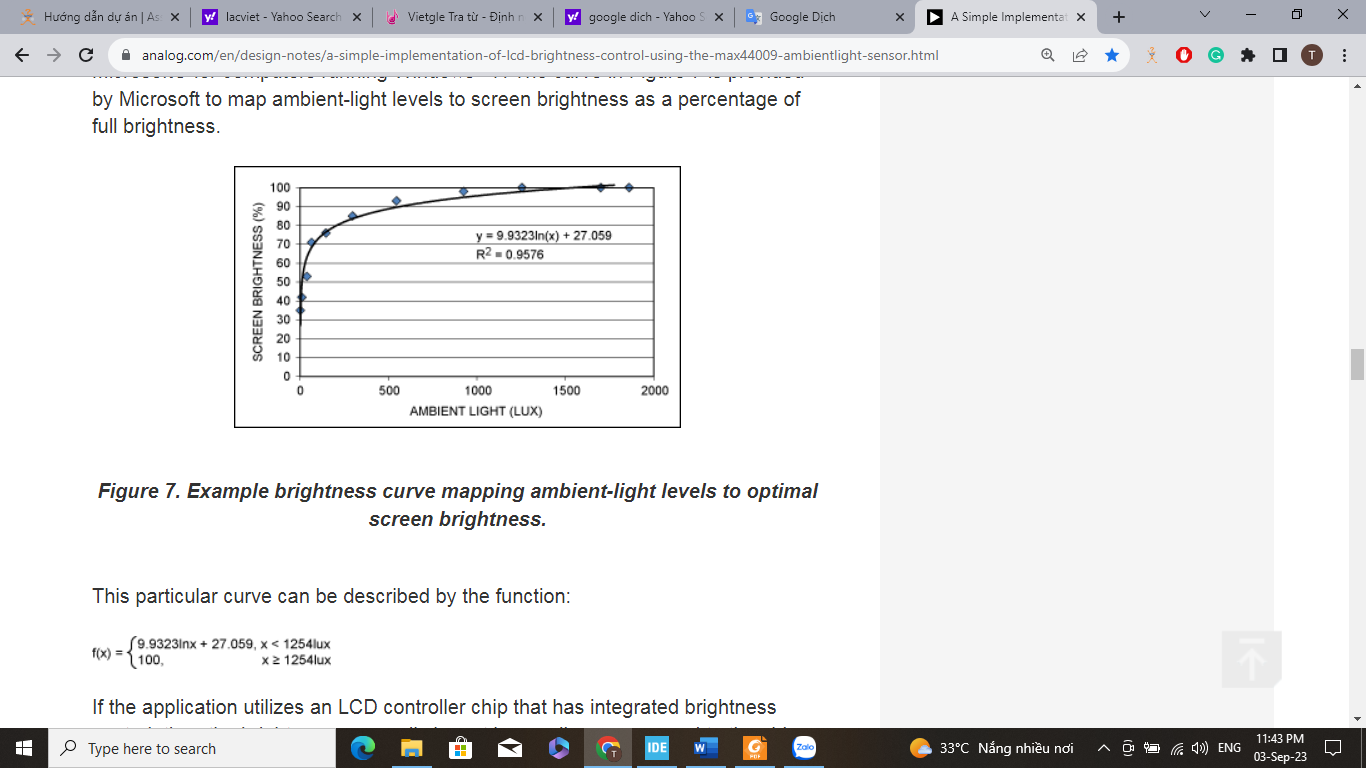
**static** uint32\_t **KalmanFilter**(uint32\_t Light\_value)

+ Lấy giá trị sau bộ lọc Kalman

ambient\_light\_value\_Kalman = KalmanFilter(ambient\_light\_value);

B3: Chuyển đổi giá trị cường độ ánh sáng sau bộ lọc Kalman thành giá trị Duty Cycle theo công thức tham khảo trong tài liệu sau:

https://www.analog.com/en/design-notes/a-simple-implementation-of-lcd-brightness-control-using-the-max44009-ambientlight-sensor.html



B4: Dựa trên giá trị Duty Cycle vừa chuyển đổi, tính toán giá trị ghi vào thanh ghi Capture/Compare của TIM1 để điều khiển độ sáng của LED

\* @func LightSensor\_AdcPollingRead

\* @brief To control the brightness of RGB

\* @param dutyCycle - the value of pulse width

\* @retval None

**static** **void** **LedControl\_TimerOCSetPwm**(uint8\_t dutyCycle)

# **Bước 5:** Tạo chu kỳ thời gian 100 ms để điều khiển giá trị độ sáng của LED.

\* @func create\_DisplayTime

\* @brief To create a fixed period to display LIGHT into LCD

\* @param None

\* @retval None

**static** **void** **create\_DisplayTime**(**void**)

**\* Thuật toán:**

B1: Xác định thời gian hiện tại.

B2: Xác định khoảng thời gian giữa thời gian khởi tạo và thời gian hiện tại

B3: Nếu khoảng thời gian đó > 100ms, thực hiện cập nhật giá trị cường độ ánh sáng của LED và điều khiển.

B4: Gán giá trị thời gian hiện tại vào thời gian khởi tạo và lặp lại các bước trên.