|  |  |
| --- | --- |
| Học trực tuyến CNTT, học lập trình từ cơ bản đến nâng cao |  |

**IOT302 – TỔNG QUAN VỀ IOT & LẬP TRÌNH C NHÚNG CHO VI ĐIỀU KHIỂN**

**THUYẾT MINH ASSIGNMENT 1**

**Công tắc cảm ứng**

|  |  |
| --- | --- |
| Tên học viên:  Mã học viên:  Ngày báo cáo: | Phạm Ngọc Hiệp  FX19756  10/02/2023 |

**HÀ NỘI, 02/2023**

**MỤC LỤC**

[1. Danh sách thư viện 1](#_heading=h.30j0zll)

[2. Danh sách các biến toàn cục và các macro 1](#_heading=h.1fob9te)

[3. Danh sách các hàm 1](#_heading=h.3znysh7)

[4. Hàm kiemtra\_tamgiac() 1](#_heading=h.2et92p0)

[5. Hàm … 2](#_heading=h.3dy6vkm)

[6. Luồng xử lý code 2](#_heading=h.1t3h5sf)

**DANH MỤC HÌNH VẼ, BẢNG BIỂU**

**NỘI DUNG THUYẾT MINH**

# Danh sách thư viện

|  |  |  |
| --- | --- | --- |
| **STT** | **Thư viện** | **Mô tả** |
| 1 | stdio.h | Thư viện có các hàm nhập xuất  + sprintf: Gửi output đã được định dạng tới một chuỗi |
| 2 | stdint.h | Thư viện lưu trữ và làm việc với các dữ liệu ở cấp độ bit  + uint8\_t : 8 bit  +uint16\_t: 16 bit  +uint32\_t: 32 bit  ….. |
| 3 | string.h | Thư viện cung cấp hàm để xử lý chuỗi |
| 4 | system\_stm32f4xx.h |  |
| 5 | timer.h |  |
| 6 | eventman.h |  |
| 7 | led.h | Thư viện cung cấp hàm để thao tác với led |
| 8 | melody.h | Thư viện cung cấp hàm để bật còi |
| 9 | lightsensor.h | Thư viện cung cấp hàm để sử dụng cảm biến ánh sáng |
| 10 | temhumsensor.h | Thư viện cung cấp hàm để sử dụng cảm biến nhiệt độ và độ ẩm |
| 11 | eventbutton.h |  |
| 12 | button.h |  |
| 13 | Ucglib.h |  |

# Danh sách các biến toàn cục và các macro

|  |  |  |
| --- | --- | --- |
| **STT** | **Biến toàn cục và macro** | **Mô tả** |
| 1 | event\_api\_t | Kiểu dữ liệu enum cho các sự kiện của chương trình |
| 2 | \*event\_api\_p | Con trỏ kiểu dữ liệu enum trỏ tới các tự kiện của chương trình |
| 3 | state\_app\_t | Kiểu dữ liệu enum cho các trạng thái của chương trình |
| 4 | eCurrentState | Biến thuộc kiểu dữ liệu enum state\_app\_t cho biết trạng thái hiện tại của chương trình |
| 5 | ucg | Biến thuộc kiểu dữ liệu struct ucgt để điều khiển màn hình LCD |
|  |  |  |
| 6 | led\_state\_b1 | Biến thuộc kiểu dữ liệu uint8\_t cho biết trạng thái hiện tại của của đèn Led khi bấm nút b1 1 lần |
| 7 | led\_state\_b2 | Biến thuộc kiểu dữ liệu uint8\_t cho biết trạng thái hiện tại của của đèn Led khi bấm nút b2 1 lần |
| 8 | led\_state\_b4 | Biến thuộc kiểu dữ liệu uint8\_t cho biết trạng thái hiện tại của của đèn Led khi bấm nút b4 1 lần |
| 9 | led\_state\_b5 | Biến thuộc kiểu dữ liệu uint8\_t cho biết trạng thái hiện tại của của đèn Led khi bấm nút b5 1 lần |
| 10 | led\_level =0; | Biến thuộc kiểu dữ liệu uint8\_t cho biết độ sáng hiện tại của led |
| 11 | state = 0; | Biến thuộc kiểu dữ liệu uint8\_t dùng để thể hiện trạng thái hiện tại của chương trình |
| 12 | ignoreSensorTime | Biến thuộc kiểu dữ liệu uint8\_t dùng để tạm hoạn thời gian bắt đầu của cảm biến |
| 13 | idTimerSensorScan = NO\_TIMER; | Biến thuộc kiểu dữ liệu SSwTimer dùng để thể hiện cho sự kiện sensor bắt đầu quét sau khi bấm nút b0 hoặc b5 5 lần |
| 14 | idTimerLedLevel = NO\_TIMER; | Biến thuộc kiểu dữ liệu SSwTimer dùng để thể hiện cho sự kiện tăng giảm độ sáng của Led khi giữ nút b1 hoặc b5 trên 1s |

# Danh sách các hàm

|  |  |  |
| --- | --- | --- |
| **STT** | **Danh sách hàm** | **Mô tả** |
| 1 | int main() | Hàm chính của chương trình |
| 2 | static void Task\_multiSensorScan(void) | Để lấy thông tin nhiệt độ, độ ẩm và hiển thị lên LCD. |
| 3 | static void SetStateApp(state\_app\_t state) | Khởi tạo trạng thái đầu tiên của chương trình khi thiết bị lần đầu được bật lên |
| 4 | static state\_app\_t GetStateApp(void) | Lấy trạng thái hiện tại của chương trình |
| 5 | static void LoadConfiguration(void) | In ra màn hình LCD dòng text khi bắt đầu cắm nguồn vào thiết bị |
| 6 | static void Increase\_LedLevel(void \*arg) | Hàm tăng độ sáng của led |
| 7 | static void Decrease\_LedLevel(void \*arg) | Hàm giảm độ sáng của led |
| 8 | void DeviceStateMachine(uint8\_t event) | Để điều khiển led,LCD, sensor với các trường hợp khác nhau khi bấm nút trên mạch |
| 9 | static void AppStateManager(uint8\_t event) | Để xử lý các sự kiện chính của chương trình |
| 10 | static void AppInitCommon() | Hàm chứa các hàm khởi tạo |

# Hàm static void Task\_multiSensorScan(void)

1. *Mô tả*

Hàm static void Task\_multiSensorScan(void) thực hiện việc lấy thông tin nhiệt độ, độ ẩm và ánh sáng sau đó hiện thị lên LCD.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Không có giá trị trả về.

# Hàm static void SetStateApp

1. *Mô tả*

Hàm static void SetStateApp(state\_app\_t state) thực hiện việc đặt trạng thái của chương trình khi lần thiết bị bật bằng cách gắn giá trị của biến eCurrentState = state.

1. *Tham số truyền vào*

Truyền vào biến state có kiểu dữ liệu enum state\_app\_t

1. *Giá trị trả về*

Không có giá trị trả về.

# Hàm static state\_app\_t GetStateApp(void)

1. *Mô tả*

Hàm state\_app\_t GetStateApp(void) thuộc kiểu giữ liệu enum state\_app\_t thực hiện việc lấy trạng thái hiện tại của chương trình và trả về trạng thái hiện tại là eCurrentState

1. *Tham số truyền vào*

*Không có tham số truyền vào*

1. *Giá trị trả về*

Trả về giá trị eCurrentState là trạng thái hiện tại của chương trình.

# Hàm static void LoadConfiguration(void)

1. *Mô tả*

Hàm static void LoadConfiguration(void) thực hiện việc hiển thị dòng text “IOT Programming by Lumi Smarthome” lên màn hình LCD của Kit mở rộng khi bắt đầu cắm nguồn cho thiết bị.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Không có giá trị trả về

# Hàm static void Increase\_LedLevel(void)

1. *Mô tả*

Hàm static void Increase\_LedLevel(void) thực hiện tăng độ sáng của đèn Led khi nhấn giữ nút B1

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Không có giá trị trả về

# Hàm static void Decrease\_LedLevel(void)

1. *Mô tả*

Hàm static void Decrease\_LedLevel(void) thực hiện giảm độ sáng của đèn Led khi nhấn giữ nút B5

1. *Tham số truyền vào*

Không có tham số truyền vào

1. *Giá trị trả về*

Không có giá trị trả về

# Hàm void DeviceStateMachine(uint8\_t event)

1. *Mô tả*

Hàm void DeviceStateMachine(uint8\_t event) thực hiện điều khiển led,LCD, sensor với các trường hợp sự kiện trạng thái khác nhau của chương trình khi bấm nút trên thiết bị.

1. *Tham số truyền vào*

Hàm truyền vào biến event với kiểu dữ liệu là uint8\_t thể hiện sự kiện khác nhau khi bấm nút trên mạch

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm static void AppStateManager(uint8\_t event)

1. *Mô tả*

Hàm static void AppStateManager(uint8\_t event) thực hiện quản lý các trạng thái của chương trình

1. *Tham số truyền vào*

Hàm truyền vào biến event với kiểu dữ liệu là uint8\_t thể hiện sự kiện khác nhau khi bấm nút trên thiết bị.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm static void AppInitCommon(void)

1. *Mô tả*

Hàm static void AppInitCommon(void) thực hiện việc khởi tạo các tài nguyên cần sử dụng

Hàm SystemCoreClockUpdate để cấu hình clock của hệ thống là 84Mhz

Hàm TimerInit để khởi tạo timer system tick để xử lý các sự kiện thời gian.

Hàm EventSchedulerInit(AppStateManager) để khởi tạo bộ đem lưu trữ danh sách sự kiện của chương trình khi chạy với tham số là con trỏ hàm app\_state\_callback để xử lý các sự kiện .

Hàm EventButton\_Init để cấu hình chân GPIO của các nút nhấn trên mạch.

Hàm BuzzerControl\_Init để cấu hình chân GPIO của còi trên mạch.

Hàm LedControl\_Init để cấu hình chân GPIO của các led RGB trên mạch.

Hàm LightSensor\_Init để cấu hình ngoại vi ADC hoạt động ở chế độ DMA để đọc dữ liệu của cảm biến ánh sáng trên mạch.

Hàm TemHumSensor\_Init để cấu hình ngoại vi I2C để giao tiếp với cảm biến nhiệt độ - độ ẩm..

Ucglib4WireSWSPI\_begin, ucg\_ClearScreen, ucg\_SetFont, ucg\_SetColor, ucg\_SetRotate180 để hiển thị dữ liệu ra màn hình LCD.

1. *Tham số truyền vào*

Không có tham số truyền vào.

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Hàm int main(void)

1. *Mô tả*

Hàm int main(void) thực hiện việc gọi hàm :

AppInitCommon() để khởi tạo các tài nguyên cần sử dụng

SetStateApp(STATE\_APP\_STARTUP) khởi tạo chương trình chính chạy đầu tiên để lấy lại trạng thái cũ của thiết bị trước khi tắt nguồn

EventSchedulerAdd(EVENT\_APP\_INIT) thêm các sự kiện khởi tạo EVENT\_APP\_INIT vào queue

Trong hàm while(1) gọi các hàm:

processTimerScheduler để xử lý các sư kiện theo thời gian đã cài đặt trước

processEventScheduler để xử lý các sự kiện khi có trạng thái ấn nút.

1. *Tham số truyền vào*

Không có tham số truyền vào

1. *Giá trị trả về*

Hàm không có giá trị trả về

# Luồng xử lý code

**Câu 1** Hiển thị dòng text “**IOT Programming by Lumi Smarthome**” lên màn hình LCD của Kit mở rộng khi bắt đầu cắm nguồn cho thiết bị.**:**

**Các bước thực xử lý:**

***Bước 1:*** Chương trình chạy hàm main :

* Chạy hàm AppInitCommon để khởi tạo các tài nguyên cần sử dụng
* Trong hàm AppInitCommon chạy hàm :
  + Hàm SystemCoreClockUpdate để cấu hình clock của hệ thống là 84Mhz
  + Hàm TimerInit để khởi tạo timer system tick để xử lý các sự kiện thời gian.
  + Hàm EventSchedulerInit(AppStateManager) để khởi tạo bộ đem lưu trữ danh sách sự kiện của chương trình khi chạy với tham số là con trỏ hàm app\_state\_callback để xử lý các sự kiện .
  + Hàm EventButton\_Init để cấu hình chân GPIO của các nút nhấn trên mạch.
  + Hàm BuzzerControl\_Init để cấu hình chân GPIO của còi trên mạch.
  + Hàm LedControl\_Init để cấu hình chân GPIO của các led RGB trên mạch.
  + Hàm LightSensor\_Init với tham số truyền vào là ADC\_READ\_MODE\_DMA để cấu hình ngoại vi ADC hoạt động ở chế độ DMA để đọc dữ liệu của cảm biến ánh sáng trên mạch.
  + Hàm TemHumSensor\_Init để cấu hình ngoại vi I2C để giao tiếp với cảm biến nhiệt độ - độ ẩm..
  + Ucglib4WireSWSPI\_begin, ucg\_ClearScreen, ucg\_SetFont, ucg\_SetColor, ucg\_SetRotate180 để hiển thị dữ liệu ra màn hình LCD.
* Chạy hàm SetStateApp với tham số truyền vào là STATE\_APP\_STARTUP để khởi tạo chương trình chính chạy đầu tiên để lấy lại trạng thái cũ của thiết bị trước khi tắt nguồn. Hàm trả về giá trị biến eCurrentState = state với state = 0
* Chạy hàm EventSchedulerADD(EVENT\_APP\_INIT) để thêm các sự kiện khởi tạo EVENT\_APP\_INIT vào queuqe.
* Trong vòng lặp while(1) chạy hàm processTimerScheduler() để xử lý các sự kiện theo thời gian đã cài đặt trước.
* Tiếp tục chạy hàm processEventScheduler() để xử lý các sự kiện khi có trạng thái ấn nút

***Bước 2 :*** Trong hàm AppStateManager với:

* Thực hiện kiểm tra eCurrentstate lấy từ hàm GetStateApp với các trường hợp
  + Trường hợp eCurrentstate bằng STATE\_APP\_STARTUP chuyển sang bước 3
  + Trường hợp eCurrentstate bằng STATE\_APP\_IDLE chuyển sang câu 2 bước 1
  + Trường hợp eCurrentstate bằng STATE\_APP\_RESET chuyển về câu 1 bước 1

***Bước 3****:* Tại trường hợp eCurrentstate bằng STATE\_APP\_STARTUP

* Kiểm tra event có bằng EVENT\_APP\_INIT không
  + Nếu sai quay trở lại bước 2
  + Nếu đúng thực hiện hàm LoadConfiguration chuyển sang bước 4 và thực hiện gọi hàm SetStateApp với tham số truyền vào STATE\_APP\_IDLE.
* Thoát khỏi câu điều kiện trở lại bước 2.

***Bước 4:*** Hàm LoadConfiguration():

* Sử dụng hàm ucg\_DrawString để hiển thị dòng chữ “IOT Programming by Lumi Smarthome” lên màn hình LCD.
* Quay trở lại bước 3

**Câu 2** Nhấn nút B3 năm lần khi đó tất cả các Led trên Kit mở rộng sẽ nháy năm lần màu GREEN với cường độ sáng là 50% và hiển thị thông tin sau thiết bị lên màn hình LCD:

*Device: Board STM32 Nucleo*

*Code: STM32F401RE\_NUCLEO*

*Manufacturer: STMicroelectronics*

*Kit expansion: Lumi Smarthome*

*Project: Simulator touch switch*

**Các bước thực xử lý:**

***Bước 1 :*** Với eCurrentState bằng STATE\_APP\_IDLE:

* Thực hiện hàm DeviceStateMachine với tham số truyền vào là event . Chuyển sang bước 2

***Bước 2:*** Tại hàm DeviceStateMachine

* Thực hiện kiểm tra giá trị event với các trường hợp:
  + Trường hợp EVENT\_OF\_BUTTON\_3\_PRESS\_5\_TIMES và EVENT\_OF\_BUTTON\_0\_PRESS\_5\_TIMES chuyển sang bước 3
  + Trường hợp EVENT\_OF\_BUTTON\_1\_PRESS\_LOGIC, EVENT\_OF\_BUTTON\_2\_PRESS\_LOGIC, EVENT\_OF\_BUTTON\_5\_PRESS\_LOGIC, EVENT\_OF\_BUTTON\_4\_PRESS\_LOGIC chuyển sang câu 3 bước 1
  + Trường hợp EVENT\_OF\_BUTTON\_1\_HOLD\_1s và EVEN\_OF\_BUTTON\_5\_HOLD\_1s chuyển sang câu 4 bước 1

***Bước 3 :*** Trường hợp EVENT\_OF\_BUTTON\_3\_PRESS\_5\_TIMES và EVENT\_OF\_BUTTON\_0\_PRESS\_5\_TIMES :

* Thực hiện hàm LedControl\_BlinkStart (LED\_ALL\_ID, BLINK\_GREEN, 10, 500, LED\_COLOR\_BLACK) để nháy led màu xanh 5 lần
* Thực hiện hàm ucg\_ClearScreen, uc\_SetFont và ucg\_DrawString để thực hiện xóa màn hình, cài font chữ hiển thị và thực hiện hiển thị thông tin theo yêu cầu của đề bài lên màn hình LCD
* Chuyển sang bước 1 câu 5.

**Câu 3** Nhấn các nút B1, B2, B5, B4 một lần để điều khiển bật/tắt màu tương ứng RED, GREEN, BLUE, WHITE của tất cả các led RGB trên Kit mở rộng và còi sẽ kêu một bíp một lần.

**Các bước thực xử lý:**

***Bước 1: Thực hiện việc kiểm tra trạng thái đèn led với nút bấm tương ứng***

* Trường hợp biến trạng thái led với nút ấn tương ứng bằng 0 (Led tại nút bấm chưa sáng)
  + Sử dụng hàm LedControl\_SetAllColor với đối số truyền vào là màu theo nút ấn và cường độ sáng là 50%
  + Sử dụng hàm BuzzerControl\_SetMelody với đối số truyền vào là pbeep trong thử viện melody.h để còi kêu một tiếng bíp.
  + Thực hiện việc gắn giá trị cho biến trạng thái led với nút ấn tương ứng bằng 1 để thể hiện trạng thái đèn đang bật.
  + Quay trở lại câu 2 bước 2
* Trường hợp biến trạng thái led với nút ấn tương ứng khác 0 (Led tại nút bấm đang được bật sáng)
  + Sử dụng hàm LedControl\_SetAllColor với đối số truyền vào là màu theo nút ấn và cường độ sáng là 0 để tắt đèn led
  + Sử dụng hàm BuzzerControl\_SetMelody với đối số truyền vào là pbeep trong thử viện melody.h để còi kêu một tiếng bíp.
  + Thực hiện việc gắn giá trị cho biến trạng thái led với nút ấn tương ứng bằng 0 để thể hiện trạng thái đèn đang tắt.
  + Quay trở lại câu 2 bước 2

**Câu 4** Nhấn giữ B1/B5 để điều khiển tăng/giảm cường độ sáng của led RGB:

* Nhấn giữ nút B1 với thời gian T lớn hơn một giây để điều khiển tăng độ sáng của led và nhà ra thì dừng điều khiển.
* Nhấn giữ nút B5 với thời gian T lớn hơn một giây để điều khiển giảm độ sáng của led và nhà ra thì dừng điều khiển

**Các bước thực xử lý:**

***Bước 1:*** Gắn giá trị cho biến ignoreSensorTime = 5 và thực hiện việc kiểm tra sự kiện tăng giảm độ sáng đèn led (idTimerLedLevel) với nút bấm tương ứng khác với NO\_TIMER

* Nếu đúng thực hiện sử dụng hàm TimerStop với tham số truyền vào là sự kiện tăng giảm độ sáng
* Nếu sai chuyển sáng bước 2

***Bước 2:*** Tăng giảm độ sáng đèn led

* Gắn giá trị biến tăng giảm độ sáng bằng với hàm TimerStart
* Trong hàm TimerStart truyền vào giá trị với tên của sự kiện hẹn giờ , thời gian hẹn giờ 20 ms , số lần lặp lại là 100 ,con trỏ hàm thực hiện là Increase\_LedLevel tương ứng với trường hợp nhấn giữ nút B1 hoặc hàm Decrease\_LedLevel tương ứng với trường hợp nhấn giữ nút B5, tham số truyền vào là NULL

***Bước 3 :*** Nhả nút bấm dừng tăng giảm độ sáng đèn led và led sáng bằng với giá trị lúc nhả tay :

* Nhả tay nút ấn tương ứng trường hợp EVENT\_OF\_BUTTON\_1\_RELEASED\_1s và EVENT\_OF\_BUTTON\_5\_RELEASED\_1s
* Thực hiện hàm TimerStop với tham số truyền vào là sự kiện tăng giảm độ sáng đèn led để dừng lại sự kiện tăng giảm độ sáng mà vẫn giữ nguyên được độ sáng đèn led lúc nhả tay.

**Câu 5** Cập nhật thông số nhiệt độ, độ ẩm và cường độ ánh sáng của các cảm biến lên màn hình LCD:

**Các bước thực hiện xử lý:**

***Bước 1*:**

* Kiểm tra giá trị của sự kiện cập nhật thông số sensor(idTimerSensorScan) không bằng giá trị NO\_TIMER hay không
* Nếu đúng thực hiện hàm TimerStop với tham số truyền vào là idTimerSensorScan để cho chương trình chưa thực hiện sự kiện cập nhật thông số sensor
* Nếu sai chuyển sang bước 2

***Bước 2:***

* Thực hiện gắn giá trị idTimerSensorScan bằng giá trị của hàm TimerStart
* Trong hàm TimerStart truyền vào giá trị với tên của sự kiện là “Task\_multiSensor”, thời gian hẹn giờ 1000ms(1s), số lần lặp lại là TIMER\_REPEAT\_FOREVER(vô tận), con trỏ hàm Task\_multiSensorScan, tham số truyền vào con trỏ hàm bằng NULL
* Chuyển sang bước 3

***Bước 3***

* Tại hàm Task\_multiSensorScan khai báo biến temperature và humidity với kiểu dữ liệu là uint8\_t để làm giá trị nhiệt độ và độ ẩm
* Khai báo biến light với kiểu dữ liệu là uint16\_t để làm giá trị đo ánh sáng
* Khai báo 3 chuỗi kiểu char với độ lớn 20 là strTemp, strHumi và starlight
* Sử dụng hàm memset để gán giá trị của cả 3 chuỗi bằng 0
* Thực hiện việc kiểm biến ignoreSensorTime có lớn hơn 0 không
* Nếu đúng thực hiện trừ giá trị của biến ignoreSensorTime đi 1 đơn vị và quay trở lại bước 1
* Nếu sai
  + Thực hiện việc gắn giá trị cho biến temperature bằng hàm TemHumSensor\_GetTemp để lấy giá trị nhiệt độ đọc từ cảm biến sau đó chia cho 100 để lấy giá trị nhiệt độ
  + Thực hiện việc gắn giá trị cho biến temperature bằng hàm TemHumSensor\_GetTemp để lấy giá trị nhiệt độ đọc từ cảm biến sau đó chia cho 100 để lấy giá trị nhiệt độ
  + Thực hiện việc gắn giá trị cho biến temperature bằng hàm TemHumSensor\_GetTemp để lấy giá trị nhiệt độ đọc từ cảm biến sau đó chia cho 100 để lấy giá trị nhiệt độ
  + Thực hiện việc gắn giá trị cho biến humidity bằng hàm TemHumSensor\_GetHumi để lấy giá trị độ ẩm đọc từ cảm biến sau đó chia cho 100 để lấy giá trị độ ẩm
  + Thực hiện việc gắn giá trị cho biến light bằng hàm LightSensor\_MeasureUseDMAMode để lấy giá trị ánh sáng
  + Kiểm tra trạng thái chương trình state có bằng 0 không. Nếu có sử dụng hàm ucg\_ClearScreen để xóa màn hình và đặt lại giá trị state = 1
  + Sử dụng ucg\_SetFont, sprint và ucg\_DrawString để lần lượt in ra giá trị nhiệt độ, độ ẩm và ánh sáng.