|  |  |
| --- | --- |
| 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 2**

**GIAO TIẾP VỚI PHẦN MỀM MÔ PHỎNG PC SIMULATOR KIT IOT**

|  |  |
| --- | --- |
| Tên học viên:  Mã học viên:  Ngày báo cáo: | **Nguyễn Hoàng Dũng**  **FX19603**  **16/01/2024** |

**HÀ NỘI, 01/2024**

**MỤC LỤC**

[1. Danh sách thư viện 5](#_Toc156468169)

[2. Danh sách các biến toàn cục và các macro 5](#_Toc156468170)

[3. Danh sách các hàm 7](#_Toc156468171)

[4. Hàm main ( ) 8](#_Toc156468172)

[4.1. Mô tả 8](#_Toc156468173)

[4.2. Tham số truyền vào 8](#_Toc156468174)

[4.3. Giá trị trả về 8](#_Toc156468175)

[5. Hàm AppInitCommon ( ) 9](#_Toc156468176)

[5.1. Mô tả 9](#_Toc156468177)

[5.2. Tham số truyền vào 9](#_Toc156468178)

[5.3. Giá trị trả về 10](#_Toc156468179)

[6. Hàm SetStateApp ( ) 10](#_Toc156468180)

[6.1. Mô tả 10](#_Toc156468181)

[6.2. Tham số truyền vào 10](#_Toc156468182)

[6.3. Giá trị trả về 10](#_Toc156468183)

[7. Hàm GetStateApp ( ) 10](#_Toc156468184)

[7.1. Mô tả 10](#_Toc156468185)

[7.2. Tham số truyền vào 10](#_Toc156468186)

[7.3. Giá trị trả về 10](#_Toc156468187)

[8. Hàm AppStateManager ( ) 10](#_Toc156468188)

[8.1. Mô tả 10](#_Toc156468189)

[8.2. Tham số truyền vào 10](#_Toc156468190)

[8.3. Giá trị trả về 10](#_Toc156468191)

[9. Hàm LoadConfiguration ( ) 10](#_Toc156468192)

[9.1. Mô tả 10](#_Toc156468193)

[9.2. Tham số truyền vào 10](#_Toc156468194)

[9.3. Giá trị trả về 10](#_Toc156468195)

[10. Hàm DeviceStateMachine ( ) 11](#_Toc156468196)

[10.1. Mô tả 11](#_Toc156468197)

[10.2. Tham số truyền vào 11](#_Toc156468198)

[10.3. Giá trị trả về 11](#_Toc156468199)

[11. Hàm Clamp ( ) 11](#_Toc156468200)

[11.1. Mô tả 11](#_Toc156468201)

[11.2. Tham số truyền vào 11](#_Toc156468202)

[11.3. Giá trị trả về 11](#_Toc156468203)

[12. Hàm Increase\_LedLevel ( ) 12](#_Toc156468204)

[12.1. Mô tả 12](#_Toc156468205)

[12.2. Tham số truyền vào 12](#_Toc156468206)

[12.3. Giá trị trả về 12](#_Toc156468207)

[13. Hàm Decrease\_LedLevel ( ) 12](#_Toc156468208)

[13.1. Mô tả 12](#_Toc156468209)

[13.2. Tham số truyền vào 12](#_Toc156468210)

[13.3. Giá trị trả về 12](#_Toc156468211)

[14. Hàm MultiSensorScan ( ) 12](#_Toc156468212)

[14.1. Mô tả 12](#_Toc156468213)

[14.2. Tham số truyền vào 12](#_Toc156468214)

[14.3. Giá trị trả về 12](#_Toc156468215)

[15. Hàm Task\_multiSensorScan ( ) 12](#_Toc156468216)

[15.1. Mô tả 12](#_Toc156468217)

[15.2. Tham số truyền vào 12](#_Toc156468218)

[15.3. Giá trị trả về 12](#_Toc156468219)

[16. Hàm LedCmdSetState ( ) 12](#_Toc156468220)

[16.1. Mô tả 12](#_Toc156468221)

[16.2. Tham số truyền vào 13](#_Toc156468222)

[16.3. Giá trị trả về 13](#_Toc156468223)

[17. Hàm BuzzerCmdSetState ( ) 13](#_Toc156468224)

[17.1. Mô tả 13](#_Toc156468225)

[17.2. Tham số truyền vào 13](#_Toc156468226)

[17.3. Giá trị trả về 13](#_Toc156468227)

[18. Hàm LcdCmdSetState ( ) 13](#_Toc156468228)

[18.1. Mô tả 13](#_Toc156468229)

[18.2. Tham số truyền vào 13](#_Toc156468230)

[18.3. Giá trị trả về 13](#_Toc156468231)

[19. Hàm ButtonCmdSetState ( ) 13](#_Toc156468232)

[19.1. Mô tả 13](#_Toc156468233)

[19.2. Tham số truyền vào 13](#_Toc156468234)

[19.3. Giá trị trả về 13](#_Toc156468235)

[20. Luồng xử lý code. 14](#_Toc156468236)

[20.1. Các bước thực xử lý 14](#_Toc156468237)

[ Câu 1: Hiển thị dòng text “IOT Programming by Lumi Smarthome” lên phần mềm mô phỏng 14](#_Toc156468238)

[ Câu 2 & 3: Nhấn các nút B1, B2, B4, B5 một lần để điều khiển bật/tắt màu tương ứng RED, GREEN, WHITE, BLUE của tất cả các led RGB trên Kit mở rộng và còi sẽ kêu một bíp một lần. Đồng thời, gửi bản tin cập nhật trạng thái màu led tương ứng, trạng thái còi lên phần mềm mô phỏng 14](#_Toc156468239)

[ Câu 4: Điều khiển bật/tắt led RGB trên thiết bị IOT KIT khi nhận được bản tin định dạng từ phần mềm mô phỏng 15](#_Toc156468240)

[ Câu 5: Điều khiển còi khi nhận được bản tin theo định dạng từ phần mềm mô phỏng 15](#_Toc156468241)

[ Câu 6: Điều khiển màn hình LCD hiển thị đoạn văn bản khi nhận được bản tin theo định dạng từ phần mềm mô phỏng 16](#_Toc156468242)

[ Câu 7: Đọc giá trị nhiệt độ, độ ẩm và cường độ ánh sáng từ các cảm biến với thời gian lấy mẫu (Tsample) một giây và gửi các thông số đó theo định dạng lên phần mềm mô phỏng để hiển thị dưới dạng đồ thị theo thời gian thực 16](#_Toc156468243)

[20.2. Lưu đồ thuật toán 18](#_Toc156468244)

**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 | Thư viện chứa thông tin hệ thống |
| 5 | timer.h | Thư viện cho phép quản lý các tác vụ theo thời gian |
| 6 | eventman.h | Thư viện chứa thông tin hệ thống |
| 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 | Thư viện cung cấp hàm để sử dụng các sự kiện của nút nhấn |
| 12 | button.h | Thư viện cung cấp hàm để sử dụng nút nhấn |
| 13 | Ucglib.h | Thư viện cung cấp hàm để cấu hình các chân giao tiếp SPI của STM32 |
| 14 | uartcmd.h | Thư viện cung cấp các hàm để khởi tạo giao tiếp Serial sử dụng giao thức UART và các hàm callback xử lý các sự kiện |
| 15 | serial.h | Thư viện cung cấp các hàm để gửi bản tin phản hồi lên HOST |

# Danh sách các biến toàn cục và các macro

|  |  |  |
| --- | --- | --- |
| **STT** | **Biến toàn cục và macro** | **Mô tả** |
|  | CYCLE\_LED\_CHANGE | Macro thể hiện chu kỳ thay đổi độ sáng LED |
|  | PERIOD\_SCAN\_MULTISENSOR | Macro thể hiện chu kỳ quét của cảm biến |
|  | event\_api\_t | Kiểu dữ liệu enum cho các sự kiện của chương trình |
|  | \*event\_api\_p | Con trỏ kiểu dữ liệu enum trỏ tới các sự kiện của chương trình |
|  | state\_app\_t | Kiểu dữ liệu enum cho các trạng thái của chương trình |
|  | ucg | Biến để điều khiển màn hình LCD |
|  | eCurrentState | Biến cho biết trạng thái hiện tại của chương trình |
|  | ledRed | Biến thể hiện trạng thái đèn LED, khi nhấn nút B1 |
|  | ledGreen | Biến thể hiện trạng thái đèn LED, khi nhấn nút B2 |
|  | ledWhite | Biến thể hiện trạng thái đèn LED, khi nhấn nút B4 |
|  | ledBlue | Biến thể hiện trạng thái đèn LED, khi nhấn nút B5 |
|  | ledChange | Biến dùng để thay đổi trạng thái LED khi nhấn giữ nút B1/B5 |
|  | idTimerStartIncrease | Biến lưu giá trị định danh ID của hàm TimerStart khi nhấn giữ nút B1 |
|  | idTimerStartDecrease | Biến lưu giá trị định danh ID của hàm TimerStart khi nhấn giữ nút B5 |
|  | idTimerDisplayLCD | Biến lưu giá trị định danh ID của hàm TimerStart khi gọi hàm MultiSensorScan |
|  | idTimerSensorUpdate | Biến lưu giá trị định danh ID của hàm TimerStart khi gọi hàm Task\_multiSensorScan |
|  | B3Count | Biến đếm số lần nhấn nút B3 khi thao tác trên phần mềm mô phỏng |
|  | temperature | Biến lưu giá trị nhiệt độ được cập nhật từ cảm biến |
|  | humidity | Biến lưu giá trị độ ẩm được cập nhật từ cảm biến |
|  | light | Biến lưu giá trị cường độ ánh sáng được cập nhật từ cảm biến |
|  | strTemp[30] | Mảng lưu giá trị nhiệt độ |
|  | strHumi[30] | Mảng lưu giá trị độ ẩm |
|  | strLight[30] | Mảng lưu giá trị cường độ ánh sáng |
|  | text[] | Mảng lưu chuỗi: IOT Programming by Lumi Smarthome |

# Danh sách các hàm

|  |  |  |
| --- | --- | --- |
| **STT** | **Danh sách hàm** | **Mô tả** |
|  | int main () | Hàm chính của chương trình |
|  | void AppInitCommon (); | Hàm khởi tạo các tài nguyên cần sử dụng |
|  | static void SetStateApp (state\_app\_t state); | Hàm SET trạng thái của chương trình |
|  | static state\_app\_t GetStateApp (); | Hàm Return trạng thái của chương trình |
|  | static void AppStateManager (uint8\_t event); | Hàm xử lý các sự kiện của chương trình chính |
|  | void LoadConfiguration (); | Hàm hiển thị thông tin khi bắt đầu cắm nguồn cho thiết bị |
|  | void DeviceStateMachine (uint8\_t event); | Hàm điều khiển LED, Buzzer và hiển thị thông tin sau khi nhấn các nút trên bộ KIT. Đồng thời, gửi các bản tin lên phần mềm mô phỏng |
|  | int Clamp (int value, int min , int max); | Hàm tạo khoảng giá trị |
|  | void Increase\_LedLevel (); | Hàm tăng độ sáng đèn LED |
|  | void Decrease\_LedLevel (); | Hàm giảm độ sáng đèn LED |
|  | void MultiSensorScan (); | Hàm cập nhật giá trị nhiệt độ, độ ẩm, cường độ ánh sáng |
|  | void Task\_multiSensorScan (); | Hàm lấy thông tin nhiệt độ, độ ẩm, cường độ ánh sáng từ cảm biến và hiển thị lên LCD. Đồng thời gửi bản tin lên phần mềm mô phỏng |
|  | void LedCmdSetState (uint8\_t led\_id, uint8\_t led\_color, uint8\_t led\_num\_blink, uint8\_t led\_interval, uint8\_t led\_last\_state); | Hàm điều khiển LED khi nhận được bản tin từ phần mềm mô phỏng |
|  | void BuzzerCmdSetState (uint8\_t buzzer\_state); | Hàm điều khiển còi khi nhận được bản tin từ phần mềm mô phỏng |
|  | void LcdCmdSetState (char \*text); | Hàm hiển thị đoạn text lên màn hình hiển thị LCD khi nhận được bản tin từ phần mềm mô phỏng |
|  | void ButtonCmdSetState (uint8\_t button\_id, uint8\_t button\_state); | Hàm điều khiển nút nhấn khi nhận được bản tin từ phần mềm mô phỏng |

# Hàm main ( )

## 4.1. Mô tả

Hàm main ( ) thực hiện việc gọi các hàm sau:

* AppInitCommon()
* SetStateApp(): Khởi tạo chương trình chính về trạng cũ của thiết bị trước khi tắt nguồn
* EventSchedulerAdd(): Thêm các sự kiện khởi tạo vào Queue
* Trong hàm while(1) gọi các hàm:
* processTimerScheduler(): Xử lý các sự kiện theo thời gian đã cài đặt trước
* processEventScheduler(): Xử lý các sự kiện khi có trạng thái nhấn nút
* processSerialReceiver(): Xử lý các bản tin nhận được theo định dạng từ phần mềm mô phỏng

## 4.2. Tham số truyền vào

Không có

## 4.3. Giá trị trả về

Không có

# Hàm AppInitCommon ( )

## 5.1. Mô tả

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

* SystemCoreClockUpdate (): Cấu hình clock của hệ thống là 84 Mhz
* TimerInit (): Khởi tạo timer system tick để xử lý các sự kiện thời gian
* EventSchedulerInit (AppStateManager): Khởi tạo bộ đệm lưu trữ danh sách sự kiện của chương trình
* EventButton\_Init (): Cấu hình chân GPIO của các nút nhấn trên mạch
* BuzzerControl\_Init (): Cấu hình chân GPIO của còi trên mạch
* LedControl\_Init (): Cấu hình chân GPIO của các led RGB trên mạch
* LightSensor\_Init (ADC\_READ\_MODE\_DMA): Cấu hình ngoại vi ADC hoạt động ở chế độ DMA để đọc dữ liệu của cảm biến ánh sáng trên mạch.
* TemHumSensor\_Init (): Cấu hình ngoại vi I2C để giao tiếp với cảm biến nhiệt độ - độ ẩm
* EventSerial\_Init (): Cấu hình ngoại vi USART ở chế độ song công (tốc độ truyền là 57600, 8 bit dữ liệu và 1 bit stop) và khởi tạo bộ đệm buffer nhận dữ liệu từ phần mềm mô phỏng Simulator
* EventSerial\_SetEventLedCallback (LedCmdSetState): Xử lý dữ liệu của bản tin điều khiển LED
* EventSerial\_SetEventButtonCallback (ButtonCmdSetState): Xử lý dữ liệu của bản tin điều khiển nút nhấn
* EventSerial\_SetEventBuzzerCallback (BuzzerCmdSetState): Xử lý dữ liệu của bản tin điều khiển còi
* EventSerial\_SetEventLcdCallback (LcdCmdSetState): Xử lý dữ liệu của bản tin điều khiển màn hình LCD
* Ucglib4WireSWSPI\_begin (&ucg, UCG\_FONT\_MODE\_SOLID): Cấu hình các chân giao tiếp SPI của STM32 với màn hình LCD
* ucg\_ClearScreen (&ucg): Xóa màn hình hiển thị LCD
* ucg\_SetFont (&ucg, ucg\_font\_ncenR10\_hf): Cài đặt font chữ kiểu ncenR10\_hr để hiển thị
* ucg\_SetColor (&ucg, 0, 255, 255, 255): SET màu trắng cho text hiển thị
* ucg\_SetColor (&ucg, 1, 0, 0, 0): SET màu đen cho nền hiển thị (backgroup)
* ucg\_SetRotate180 (&ucg): Xoay màn hình 1 góc 180 độ

## 5.2. Tham số truyền vào

Không có

## 5.3. Giá trị trả về

Không có

# Hàm SetStateApp ( )

## 6.1. Mô tả

Hàm SetStateApp ( ) thực hiện việc đặt trạng thái của chương trình khi bật thiết bị

## 6.2. Tham số truyền vào

Biến state (kiểu dữ liệu enum: state\_app\_t)

## 6.3. Giá trị trả về

Không có

# Hàm GetStateApp ( )

## 7.1. Mô tả

Hàm GetStateApp ( ) thực hiện việc lấy trạng thái hiện tại của chương trình

## 7.2. Tham số truyền vào

Không có

## 7.3. Giá trị trả về

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

# Hàm AppStateManager ( )

## 8.1. Mô tả

Hàm AppStateManager ( ) thực hiện việc xử lý các sự kiện của chương trình chính

## 8.2. Tham số truyền vào

Biến event (kiểu dữ liệu uint8\_t): Sự kiện khi bấm nút trên bộ KIT

## 8.3. Giá trị trả về

Không có

# Hàm LoadConfiguration ( )

## 9.1. Mô tả

Hàm LoadConfiguration ( ) 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.

## 9.2. Tham số truyền vào

Không có

## 9.3. Giá trị trả về

Không có

# Hàm DeviceStateMachine ( )

## 10.1. Mô tả

Hàm DeviceStateMachine ( ) thực hiện việc điều khiển Led, Buzzer và hiển thị thông tin sau khi nhấn các nút trên bộ KIT. Đồng thời, gửi bản tin cập nhật trạng thái màu led tương ứng, trạng thái còi lên phần mềm mô phỏng

* Nhấn nút B3 năm lần khi đó tất cả các Led trên Kit mở rộng sẽ nháy năm lần màu GREEN 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*

Sau khi hiển thị các thông tin trên, thì hiển thị thông tin nhiệt độ, độ ẩm, cường độ ánh sáng được cập nhật từ cảm biến lên màn hình LCD

* Nhấn các nút B1, B2, B4, B5 một lần để điều khiển bật/tắt màu tương ứng RED, GREEN, WHITE, BLUE của tất cả các led RGB trên Kit mở rộng và còi sẽ kêu một bíp một lần. Đồng thời, gửi bản tin cập nhật trạng thái màu led tương ứng, trạng thái còi kêu một tiếng bíp lên phần mềm mô phỏng
* 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 nhả 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 nhả ra thì dừng điều khiển.

## 10.2. Tham số truyền vào

Biến event (kiểu dữ liệu uint8\_t): Sự kiện khi bấm nút trên bộ KIT

## 10.3. Giá trị trả về

Không có

# Hàm Clamp ( )

## 11.1. Mô tả

Hàm Clamp ( ) thực hiện việc tạo khoảng giá trị của biến

## 11.2. Tham số truyền vào

* Biến value (kiểu dữ liệu int): Giá trị hiện tại
* Biến min (kiểu dữ liệu int): Giá trị thấp nhất
* Biến max (kiểu dữ liệu int): Giá trị cao nhất

## 11.3. Giá trị trả về

Trả về giá trị value

# Hàm Increase\_LedLevel ( )

## 12.1. Mô tả

Hàm Increase\_LedLevel ( ) thực hiện việc tăng độ sáng đèn LED khi nhấn giữ nút B1

## 12.2. Tham số truyền vào

Không có

## 12.3. Giá trị trả về

Không có

# Hàm Decrease\_LedLevel ( )

## 13.1. Mô tả

Hàm Decrease\_LedLevel ( ) thực hiện việc giảm độ sáng đèn LED khi nhấn giữ nút B5

## 13.2. Tham số truyền vào

Không có

## 13.3. Giá trị trả về

Không có

# Hàm MultiSensorScan ( )

## 14.1. Mô tả

Hàm MultiSensorScan ( ) thực hiện việc cập nhật giá trị nhiệt độ, độ ẩm, cường độ ánh sáng

## 14.2. Tham số truyền vào

Không có

## 14.3. Giá trị trả về

Không có

# Hàm Task\_multiSensorScan ( )

## 15.1. Mô tả

Hàm Task\_multiSensorScan ( ) thực hiện việc lấy giá trị nhiệt độ, độ ẩm, cường độ ánh sáng từ cảm biến và hiển thị lên LCD. Đồng thời, gửi các giá trị đó lên phần mềm mô phỏng

## 15.2. Tham số truyền vào

Không có

## 15.3. Giá trị trả về

Không có

# Hàm LedCmdSetState ( )

## 16.1. Mô tả

Hàm LedCmdSetState ( ) thực hiện việc điều khiển bật/tắt led RGB trên thiết bị IOT KIT khi nhận được bản tin từ phần mềm mô phỏng

## 16.2. Tham số truyền vào

* Biến led\_id (kiểu dữ liệu uint8\_t): LED tương ứng trên KIT
* Biến led\_color (kiểu dữ liệu uint8\_t): Màu sắc tương ứng của LED
* Biến led\_num\_blinked\_id (kiểu dữ liệu uint8\_t): Số lần bật tắt LED
* Biến led\_interval (kiểu dữ liệu uint8\_t): Chu kỳ bật tắt của LED
* Biến led\_last\_state (kiểu dữ liệu uint8\_t): Trạng thái cuối của LED

## 16.3. Giá trị trả về

Không có

# Hàm BuzzerCmdSetState ( )

## 17.1. Mô tả

Hàm BuzzerCmdSetState ( ) thực hiện việc điều khiển còi khi nhận được bản tin từ phần mềm mô phỏng

## 17.2. Tham số truyền vào

Biến buzzer\_state (kiểu dữ liệu uint8\_t): Trạng thái của còi

## 17.3. Giá trị trả về

Không có

# Hàm LcdCmdSetState ( )

## 18.1. Mô tả

Hàm LcdCmdSetState ( ) thực hiện việc hiển thị đoạn text lên màn hình hiển thị LCD khi nhận được bản tin từ phần mềm mô phỏng

## 18.2. Tham số truyền vào

Con trỏ text (kiểu dữ liệu char): Đoạn text cần hiển thị

## 18.3. Giá trị trả về

Không có

# Hàm ButtonCmdSetState ( )

## 19.1. Mô tả

Hàm ButtonCmdSetState ( ) thực hiện việc điều khiển nút nhấn khi nhận được bản tin từ phần mềm mô phỏng

## 19.2. Tham số truyền vào

* Biến button\_id (kiểu dữ liệu uint8\_t): Nút nhấn tương ứng trên KIT
* Biến button\_state (kiểu dữ liệu uint8\_t): Trạng thái của nút nhấn

## 19.3. Giá trị trả về

Không có

# Luồng xử lý code.

## 20.1. Các bước thực xử lý

* **Câu 1:** Hiển thị dòng text “**IOT Programming by Lumi Smarthome**” lên phần mềm mô phỏng
* Bước 1: Chương trình chạy hàm main
* Gọi hàm AppInitCommon để khởi tạo các tài nguyên cần sử dụng
* Gọi hàm SetStateApp với tham số truyền vào là STATE\_APP\_STARTUP: Khởi tạo chương trình chính về trạng thái cũ của thiết bị trước khi tắt nguồn. Hàm trả về giá trị biến eCurrentState = state với state = 0
* Gọi hàm EventSchedulerAdd với tham số truyền vào là EVENT\_APP\_INIT để thêm các sự kiện khởi tạo EVENT\_APP\_INIT vào Queue
* Trong vòng lặp while(1) gọi các hàm:
* processTimerScheduler () để xử lý các sự kiện theo thời gian đã cài đặt trước
* processEventScheduler () để xử lý các sự kiện khi có trạng thái nhấn nút
* processSerialReceiver () để xử lý các bản tin nhận được theo định dạng từ phần mềm mô phỏng
* Bước 2: Trong hàm AppStateManager
* 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: Trường hợp eCurrentState bằng STATE\_APP\_STARTUP
* Kiểm tra event có bằng EVENT\_APP\_INIT hay không:
* Nếu đúng:
* Gọi hàm LoadConfiguration để hiển thị dòng chữ **“IOT Programming by Lumi Smarthome”** lên màn hình LCD
* Thực hiện hàm Serial\_SendPacket để hiển thị dòng chữ **“IOT Programming by Lumi Smarthome”** lên phần mềm mô phỏng
* Gọi hàm SetStateApp với tham số truyền vào là STATE\_APP\_IDLE
* Thoát khỏi câu điều kiện và quay trở lại bước 2
* Nếu sai: Quay trở lại bước 2
* **Câu 2 & 3:** Nhấn các nút B1, B2, B4, B5 một lần để điều khiển bật/tắt màu tương ứng RED, GREEN, WHITE, BLUE của tất cả các led RGB trên Kit mở rộng và còi sẽ kêu một bíp một lần. Đồng thời, gửi bản tin cập nhật trạng thái màu led tương ứng, trạng thái còi lên phần mềm mô phỏng
* Thực hiện kiểm tra trạng thái đèn Led theo các nút bấm tương ứng
* Trường hợp biến trạng thái đèn Led với nút nhấn tương ứng bằng 0 (đèn Led đang không sáng)
* Gán biến trạng thái đèn Led với nút nhấn tương ứng bằng 1 để thể hiện trạng thái đèn Led đang bật, đồng thời gán trạng thái các Led khác bằng 0 để đặt lại trạng thái các đèn Led đó
* Thực hiện hàm LedControl\_SetAllColor để bật đèn Led. Đối số truyền vào là màu đèn Led theo nút nhấn tương ứng và cường độ sáng là 50%
* Thực hiện hàm LedControl\_SendPacketRespond để gửi bản tin cập nhật trạng thái màu Led tương ứng lên phần mềm mô phỏng. Đối số truyền vào là Led tương ứng trên KIT, màu Led theo nút nhấn và cường độ ánh sáng là 50%
* Thực hiện hàm BuzzerControl\_SetMelody để bật còi kêu 1 tiếng bíp. Đối số truyền vào là pbeep
* Thực hiện hàm BuzzerControl\_SendPacketRespond để gửi bản tin cập nhật trạng thái còi lên phần mềm mô phỏng. Đối số trường vào là 1
* Trường hợp trạng thái đèn Led với nút nhấn tương ứng bằng 1 (đèn LED đang sáng)
* Gán biến trạng thái đèn Led với nút nhấn tương ứng bằng 0 để thể hiện trạng thái đèn Led đang tắt
* Thực hiện hàm LedControl\_SetAllColor để tắt đèn Led. Đối số truyền vào là màu đèn Led theo nút nhấn tương ứng và cường độ sáng là 0%
* Thực hiện hàm LedControl\_SendPacketRespond để gửi bản tin cập nhật trạng thái màu Led tương ứng lên phần mềm mô phỏng. Đối số truyền vào là Led tương ứng trên KIT, màu Led theo nút nhấn và cường độ ánh sáng là 0%
* Thực hiện hàm BuzzerControl\_SetMelody để bật còi kêu 1 tiếng bíp. Đối số truyền vào là pbeep
* Thực hiện hàm BuzzerControl\_SendPacketRespond để gửi bản tin cập nhật trạng thái còi lên phần mềm mô phỏng. Đối số trường vào là 1
* **Câu 4:** **Điều khiển bật/tắt led RGB trên thiết bị IOT KIT khi nhận được bản tin định dạng từ phần mềm mô phỏng**
* Kiểm tra giá trị của led\_id với địa chỉ của các Led tương ứng trên KIT
* Kiểm tra led\_last\_state với các màu Led tương ứng trên giao diện của phần mềm mô phỏng (RED, GREEN, BLUE)
* Thực hiện hàm LedControl\_SetColorGeneral để bật/tắt các Led tương ứng
* **Câu 5: Điều khiển còi khi nhận được bản tin theo định dạng từ phần mềm mô phỏng**
* Kiểm tra giá trị của buzzer\_state với trạng thái của còi
* Thực hiện hàm BuzzerControl\_SetMelody để bật/tắt còi kêu 1 tiếng bíp
* **Câu 6: Điều khiển màn hình LCD hiển thị đoạn văn bản khi nhận được bản tin theo định dạng từ phần mềm mô phỏng**
* Bước 1: So sánh idTimerSensorUpdate với NO\_TIMER
* Nếu idTimerSensorUpdate không bằng NO\_TIMER
* Thực hiện hàm TimerStop để dừng sự kiện. Đối số truyền vào là idTimerSensorUpdate
* Gán idTimerSensorUpdate bằng NO\_TIMER
* Bước 2: Thực hiện in đoạn text từ phần mềm mô phỏng ra màn hình LCD
* **Câu 7:** **Đọc giá trị nhiệt độ, độ ẩm và cường độ ánh sáng từ các cảm biến với thời gian lấy mẫu (Tsample) một giây và gửi các thông số đó theo định dạng lên phần mềm mô phỏng để hiển thị dưới dạng đồ thị theo thời gian thực**
* Bước 1: So sánh idTimerDisplayLCD với NO\_TIMER
* Nếu idTimerDisplayLCD không bằng NO\_TIMER
* Thực hiện hàm TimerStop để dừng sự kiện. Đối số truyền vào là idTimerDisplayLCD
* Gán idTimerDisplayLCD bằng NO\_TIMER
* Bước 2: Gọi hàm MultiSensorScan
* Gán idTimerDisplayLCD bằng giá trị trả về của hàm TimerStart
* Đối số truyền vào hàm TimerStart (trong trường hợp này):
* Đối số 1: Tên của sự kiện hẹn giờ (MultiSensorScan)
* Đối số 2: Thời gian hẹn giờ (7000 ms)
* Đối số 3: Số lần lặp lại (1 lần)
* Đối số 4: Con trỏ hàm thực hiện (hàm MultiSensorScan)
* Đối số 5: Tham số truyền vào con trỏ hàm (NULL)
* Bước 3: Thực hiện hàm MultiSensorScan
* Thực hiện hàm ucg\_ClearScreen, ucg\_SetFont để xóa màn hình và cài đặt font chữ hiển thị lên màn hình LCD
* So sánh idTimerSensorUpdate với NO\_TIMER
* Nếu idTimerSensorUpdate không bằng NO\_TIMER
* Thực hiện hàm TimerStop để dừng sự kiện. Đối số truyền vào là idTimerSensorUpdate
* Gán idTimerSensorUpdate bằng NO\_TIMER
* Bước 4: Gọi hàm Task\_multiSensorScan
* Gán idTimerSensorUpdate bằng giá trị trả về của hàm TimerStart
* Đối số truyền vào hàm TimerStart (trong trường hợp này):
* Đối số 1: Tên của sự kiện hẹn giờ (Task\_multiSensorScan)
* Đối số 2: Thời gian hẹn giờ (PERIOD\_SCAN\_MULTISENSOR: 1000 ms)
* Đối số 3: Số lần lặp lại (TIMER\_REPEAT\_FOREVER: lặp vô hạn)
* Đối số 4: Con trỏ hàm thực hiện (hàm Task\_multiSensorScan)
* Đối số 5: Tham số truyền vào con trỏ hàm (NULL)
* Bước 5: Thực hiện hàm Task\_multiSensorScan
* Lấy giá trị nhiệt độ, độ ẩm và cường độ ánh sáng từ cảm biến
* Thực hiện hàm TemHumSensor\_GetTemp để lấy giá trị nhiệt độ từ cảm biến, sau đó chia cho 100 để lấy giá trị nhiệt độ và gán vào biến temperature
* Thực hiện hàm TemHumSensor\_GetHumi để lấy giá trị độ ẩm từ cảm biến, sau đó chia cho 100 để lấy giá trị độ ẩm và gán vào biến humidity
* Thực hiện hàm LightSensor\_MeasureUseDMAMode để lấy giá trị cường độ ánh sáng từ cảm biến và gán vào biến light
* Lưu giá trị nhiệt độ, độ ẩm và cường độ ánh sáng vào các mảng tương ứng strTemp, strHumi, strLight
* Thực hiện hàm memset để gán giá trị của cả 3 mảng bằng 0
* Thực hiện hàm sprintf để ghi các giá trị đã được định dạng vào các mảng
* Thực hiện hàm ucg\_DrawString để hiển thị giá trị nhiệt độ, độ ẩm và cường độ ánh sáng ra màn hình LCD
* Thực hiện hàm TempSensor\_SendPacketRespond để gửi giá trị nhiệt độ lên phần mềm mô phỏng. Đối số truyền vào là temperature
* Thực hiện hàm HumiSensor\_SendPacketRespond để gửi giá trị độ ẩm lên phần mềm mô phỏng. Đối số truyền vào là humidity
* Thực hiện hàm LightSensor\_SendPacketRespond để gửi giá trị cường độ ánh sáng lên phần mềm mô phỏng. Đối số truyền vào là light

## 20.2. Lưu đồ thuật toán

