**TỔNG QUAN CÁC BƯỚC**

**Bước 1:** Khai báo các thư viện; định nghĩa macro để sử dụng **USART**, **LED, BUZZER; cảm biến nhiệt độ, độ ẩm, ánh sáng…**

**Bước 2:** Khởi tạo các tham số để sử dụng ngoại vi. Khởi tạo giao tiếp giữa USART với PC Simulator.

**Bước 3:** Lưu Frame dữ liệu nhận được khi có sự kiện trên PC Simulator vào hàng đợitrong trình phục vụ ngắt.

**Bước 4:** Lấy dữ liệu trong hàng đợi để điều khiển các thiết bị ngoại vi trên Kit mở rộng.

**Bước 5:** Dựa trên giá trị phản hồi khi nhận dữ liệu USART, đưa mảng dữ liệu nhận được trong **Bước 4** đi xử lý.

**Bước 6:** So sánh các phần tử trong mảng dữ liệu lấy ra từ hàng đợi và điều khiển LED, BUZZER trên Kit mở rộng.

**Bước 7:** Tạo Frame dữ liệu truyền, xây dựng hàm truyền và truyền giá trị của cảm biến từ MCU lên PC Simulator.

**THUYẾT MINH**

**Bước 1:** Khai báo các thư viện; định nghĩa macro để sử dụng **USART**, **LED, BUZZER; cảm biến nhiệt độ, độ ẩm, ánh sáng…**

**Bước 2:** Khởi tạo các tham số để sử dụng ngoại vi. Khởi tạo giao tiếp giữa USART với PC Simulator.

**1. Khởi tạo các tham số để sử dụng ngoại vi**

/\*

\* @func AppInitCommon

\* @brief To initialize all peripherals: SystemClock, TIMER, BUZZER, LED, Sensor

\* @param None

\* @retval None

\*/

**static** **void** **AppInitCommon**(**void**)

**2. Khởi tạo giao tiếp giữa USART với PC Simulator**

/\*

\* @func Serial\_Init

\* @brief To initialize FIFO buffer and USART2

\* @param None

\* @retval None

\*/

**void** **SerialInit**(**void**)

**Bước 3:** Lưu Frame dữ liệu nhận được khi có sự kiện trên PC Simulator vào hàng đợitrong trình phục vụ ngắt.

/\*

\* @func USART2\_IRQHandler

\* @brief To process the interrupt of USART2

\* @param None

\* @retval None

\*/

**void** **USART2\_IRQHandler**(**void**)

**\* Thuật toán:**

B1: Khi cờ ngắt nhận dữ liệu RXNE of USART2 được thiết lập .

B2: Đọc dữ liệu trong thanh ghi USART\_DR.

B3: Sử dụng hàm bufEnDat để đưa dữ liệu vào trong hàng đợi.

**Bước 4:** Lấy dữ liệu trong hàng đợi để điều khiển các thiết bị ngoại vi trên Kit mở rộng.

/\*

\* @func PollRxBuff

\* @brief To process all received news

\* @param None

\* @retval The state of USART2

\*/

**static** uint8\_t **PollRxBuff**(**void**)

**\* Thuật toán:**

B1: Sử dụng vòng lặp **while** để kiểm tra hàng đợi có còn dữ liệu hay không và USART có đang ở trạng thái rảnh hay không.

B2: Nếu không thỏa mãn thoát vòng lặp **while**. Nếu thỏa mãn, lấy byte dữ liệu ra khỏi hàng đợi 🡪 sử dụng câu lệnh **switch** kiểm tra byte nhận được với **Byte Start, Byte Data, Byte CheckXor** và đưa **Byte Data** vào mảng dữ liệu nhận. Lặp lại bước này cho đến khi xử lý xong **Frame** được gửi xuống từ PC Simulator.

B3: Trả lại trạng thái USART.

**Bước 5:** Dựa trên giá trị phản hồi khi nhận dữ liệu USART, đưa mảng dữ liệu nhận được trong **Bước 4** đi xử lý.

/\*

\* @func processSerialReceiver

\* @brief To check the received data

\* @param None

\* @retval None

\*/

**void** **process\_Serial\_Receiver**(**void**)

**\* Thuật toán:**

B1: Sử dụng câu lệnh switch để kiểm tra các trạng thái trả về của USART.

B2: Nếu trạng thái trả về của USART là *UART\_STATE\_DATA\_RECEIVED* 🡪 Gọi hàm xử lý dữ liệu từ mảng nhận được.

**Bước 6:** So sánh các phần tử trong mảng dữ liệu lấy ra từ hàng đợi và điều khiển LED, BUZZER trên Kit mở rộng.

/\*

\* @func UartCommandProcess

\* @brief To control LED, BUZZER

\* @param Received data array

\* @retval None

\*/

**void** **UartCommandProcess**(**void** \*arg)

**\* Thuật toán:** Lấy các trường bit trong mảng dữ liệu nhận được như CmdId, numID, color, state để điều khiển LED, BUZZER tương ứng.

**\* Hàm điều khiển LED, BUZZER**

\* @func LedBuzzerControl\_SetState

\* @brief To turn on/off LED, BUZZER following state

\* @param

\* \*GPIOx - Address of GPIO port

\* GPIO\_PIN - The pin ordinal number of GPIO port

\* Status - ON/OFF LED, BUZZER

\* @retval None

\*/

**static** **void** **LedBuzzerControl\_SetState**(GPIO\_TypeDef \*GPIOx, uint8\_t GPIO\_PIN, uint8\_t Status)

**Bước 7:** Tạo Frame dữ liệu truyền, xây dựng hàm truyền và truyền giá trị của cảm biến từ MCU lên PC Simulator.

**1. Tạo Frame dữ liệu truyền, xây dựng hàm truyền**

/\*\*

\* @func SendDataPacketIntoPC

\* @brief To send the light value of sensor into PC Simulator

\* @param

\* DataPacketLength - The length of frame

\* Option - To reserve for the furture

\* CmdId - Bulletin identifier

\* Type - Bulletin type

\* \*Data - The value array of sensor

\* LengthData- The length of that array

\* Sequence - The number of exchanged messages

\* @retval

\*/

**static** **void** **SendDataPacketIntoPC**(

uint8\_t DataPacketLength,

uint8\_t Option,

uint8\_t CmdId,

uint8\_t Type,

uint8\_t \*Data,

uint8\_t LengthData,

uint8\_t Sequence

)

**\* Thuật toán:**

B1: Đưa các trường dữ liệu theo thứ tự **Byte Start (0xB1) 🡪 DataPacketLength 🡪 Option 🡪 CmdId 🡪 Type 🡪 Data (n bytes) 🡪 Sequence 🡪 CXOR** vào một mảng để gửi dữ liệu lên PC Simulator.

B2: Truyền lần lượt từng phần tử của mảng đó lên PC Simulator.

**2. Truyền giá trị của cảm biến từ MCU lên PC Simulator**

**\* Thuật toán:**

B1: Lấy dữ liệu cảm biến

B2: Gọi hàm truyền dữ liệu và truyền

/\*\*

\* @func GetSensorValue\_Transmit

\* @brief To get the sensor value and transmit to PC Simulator Kit IOT by USART2

\* @param None

\* @retval None

\*/

**static** **void** **GetSensorValue\_Transmit**(**void**)

\* **Truyền dữ liệu nhiệt độ**

/\*\*

\* @func SendTempIntoPC

\* @brief To send the temperature value of sensor into PC Simulator

\* @param Temperature value

\* @retval None

\*/

**static** **void** **SendTempIntoPC**(uint16\_t temp)

**\* Truyền dữ liệu độ ẩm**

/\*\*

\* @func SendHumIntoPC

\* @brief To send the humidity value of sensor into PC Simulator

\* @param Humidity value

\* @retval None

\*/

**static** **void** **SendHumIntoPC**(uint16\_t hum)

**\* Truyền dữ liệu cường độ ánh sáng**

/\*\*

\* @func SendLightIntoPC

\* @brief To send the light value of sensor into PC Simulator

\* @param Light value

\* @retval None

\*/

**static** **void** **SendLightIntoPC**(uint16\_t light)