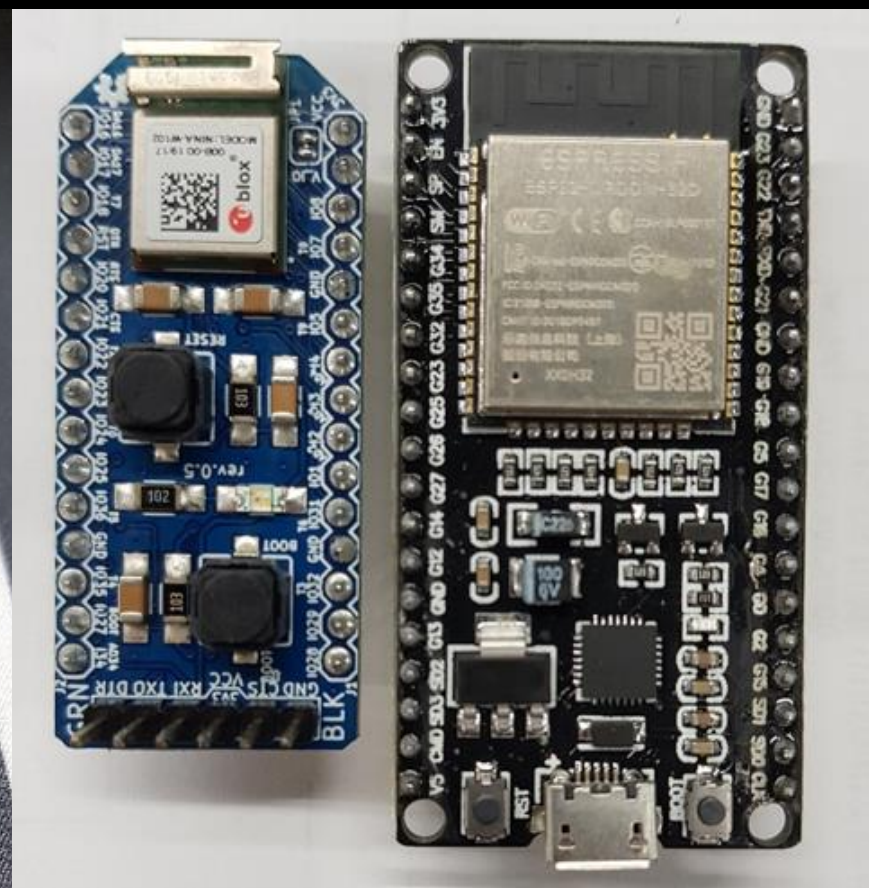


중급과정

# ESP32

## 온라인 워크숍



# 과정 목차

## ❖ 3일차

1. ESP32 Bluetooth Low Energy
  - > Architecture
  - > GATT Server
  - > SMP
2. ESP32 integration
  - > ESP32 Blufi (Ble + WIFI)

# 1. Bluetooth Low Energy

## 1.1. GAP

### 1.1.1. Overview

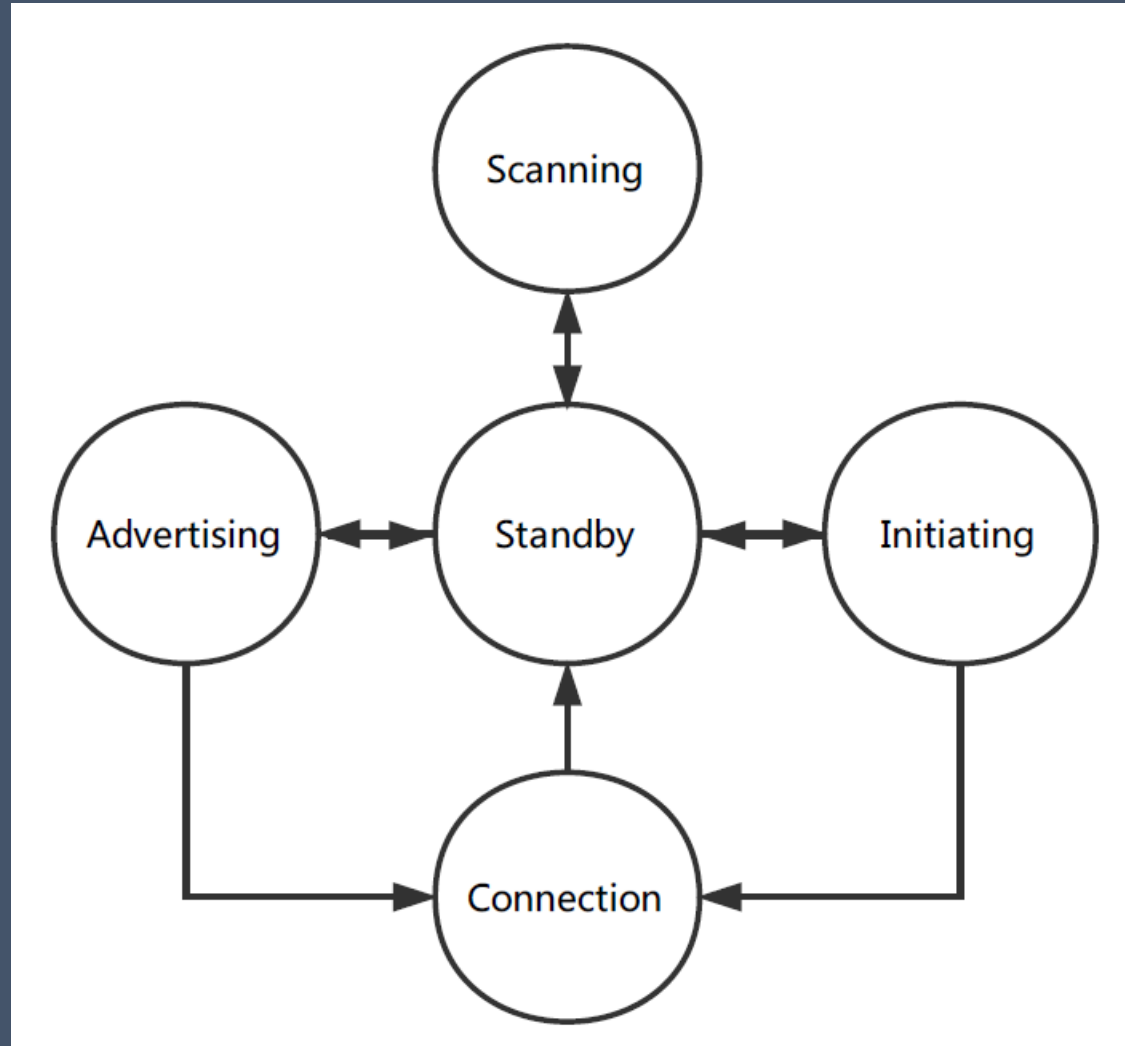
이 섹션에서는 주로 ESP32에서 BLE GAP API의 구현 및 사용에 대해 소개합니다. GAP (Generic Access Profile)는 발견(discovery) 프로세스, 장치 관리 및 BLE 장치 간의 장치 연결 설정을 정의합니다.

BLE GAP는 API 호출 및 이벤트 리턴 형식으로 구현됩니다. 프로토콜 스택에서 API 호출의 처리 결과는 이벤트에 의해 리턴됩니다. 피어 장치가 요청을 시작하면 해당 피어 장치의 상태도 이벤트에 의해 반환됩니다.

BLE 장치에 대해 정의된 4개의 GAP 역할이 있습니다.

- **Broadcaster(브로드 캐스터)** : 브로드 캐스터는 advertising 패킷을 전송하는 장치이므로 observer(관찰자)가 검색할 수 있습니다. 이 장치는 advertising만 할 수 있지만 연결할 수는 없습니다.
- **Observer(관찰자)** : Observer는 브로드 캐스터를 검색하고 이 정보를 응용 프로그램에 보고하는 장치입니다. 이 장치는 스캔 요청만 보낼 수 있지만 연결할 수는 없습니다.
- **Peripheral(주변 장치)** : Peripheral은 연결 가능한 advertising packets을 사용하여 알리고 연결되면 슬레이브가 되는 장치입니다.
- **Central(중앙)** : Central은 주변 장치에 대한 연결을 시작하고 물리적 링크가 설정되면 마스터가 되는 장치입니다.

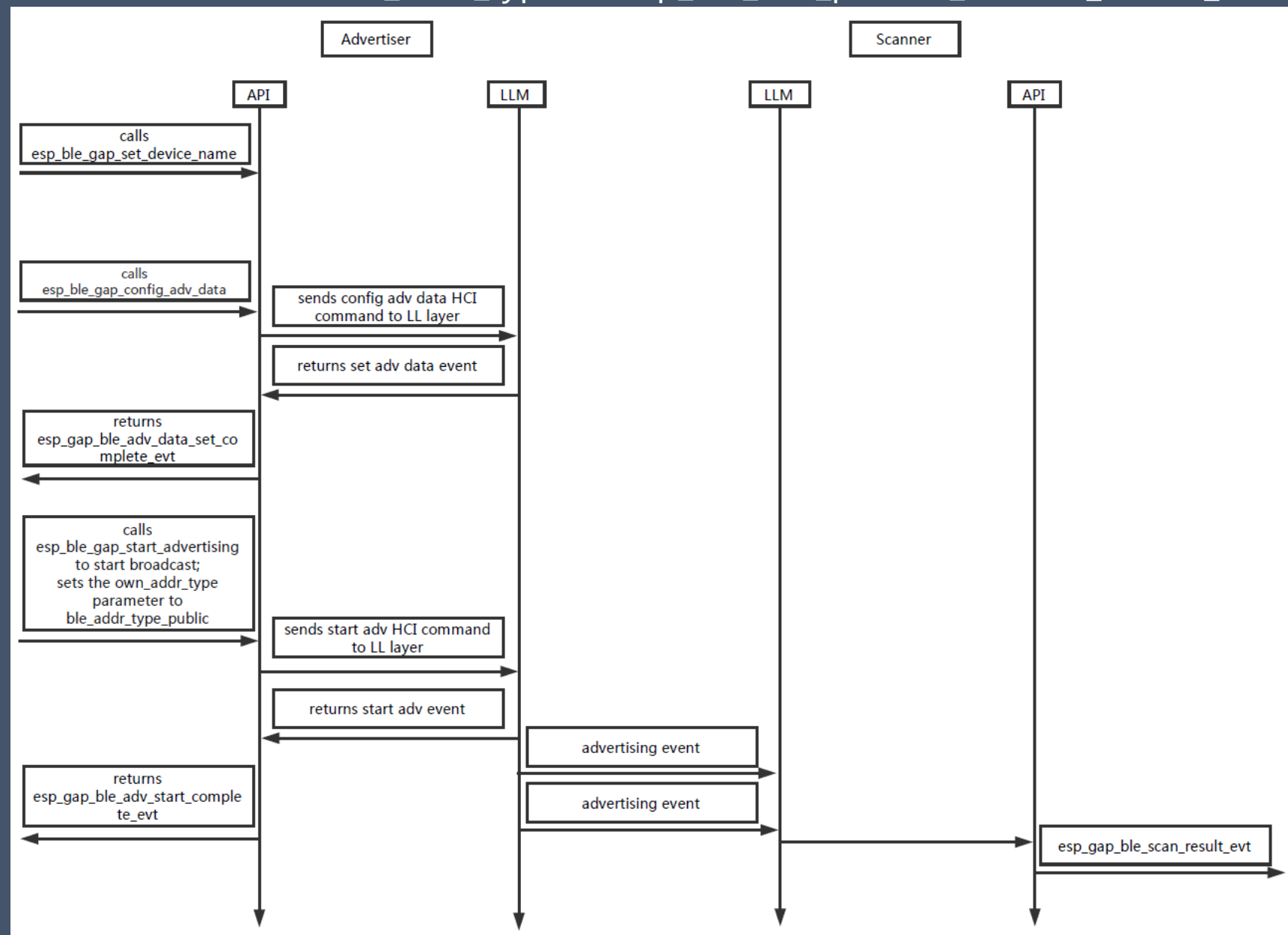
### 1.1.2. Status Transitions(전환) among GAP Roles



## 1.1.3. BLE Broadcast Procedure

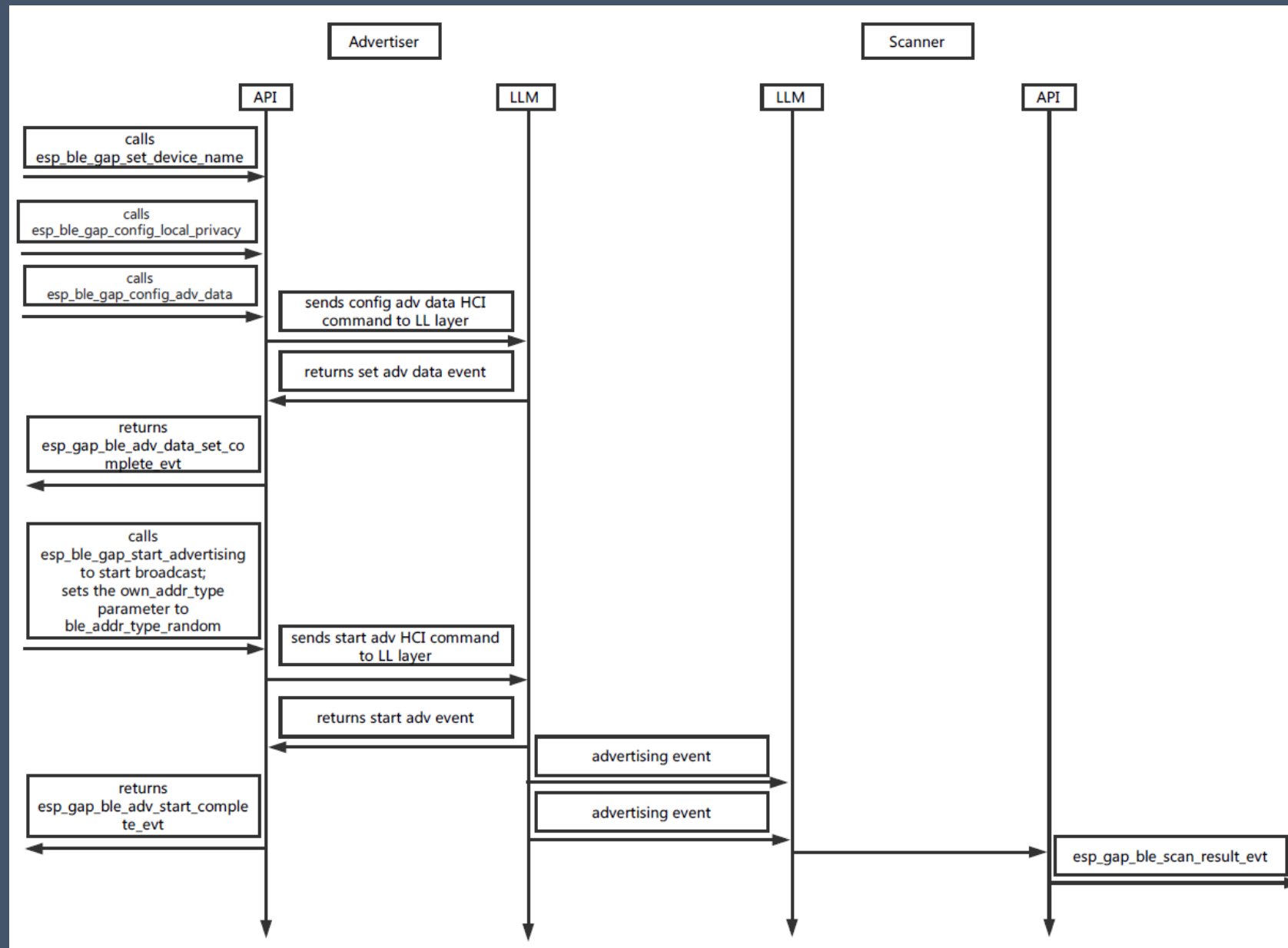
### 1.1.3.1. Broadcast using a public address

공개 주소가 브로드 캐스트에 사용될 때, own\_addr\_type의 esp\_ble\_adv\_params\_t는 **BLE\_ADDR\_TYPE\_PUBLIC**으로 설정해야합니다



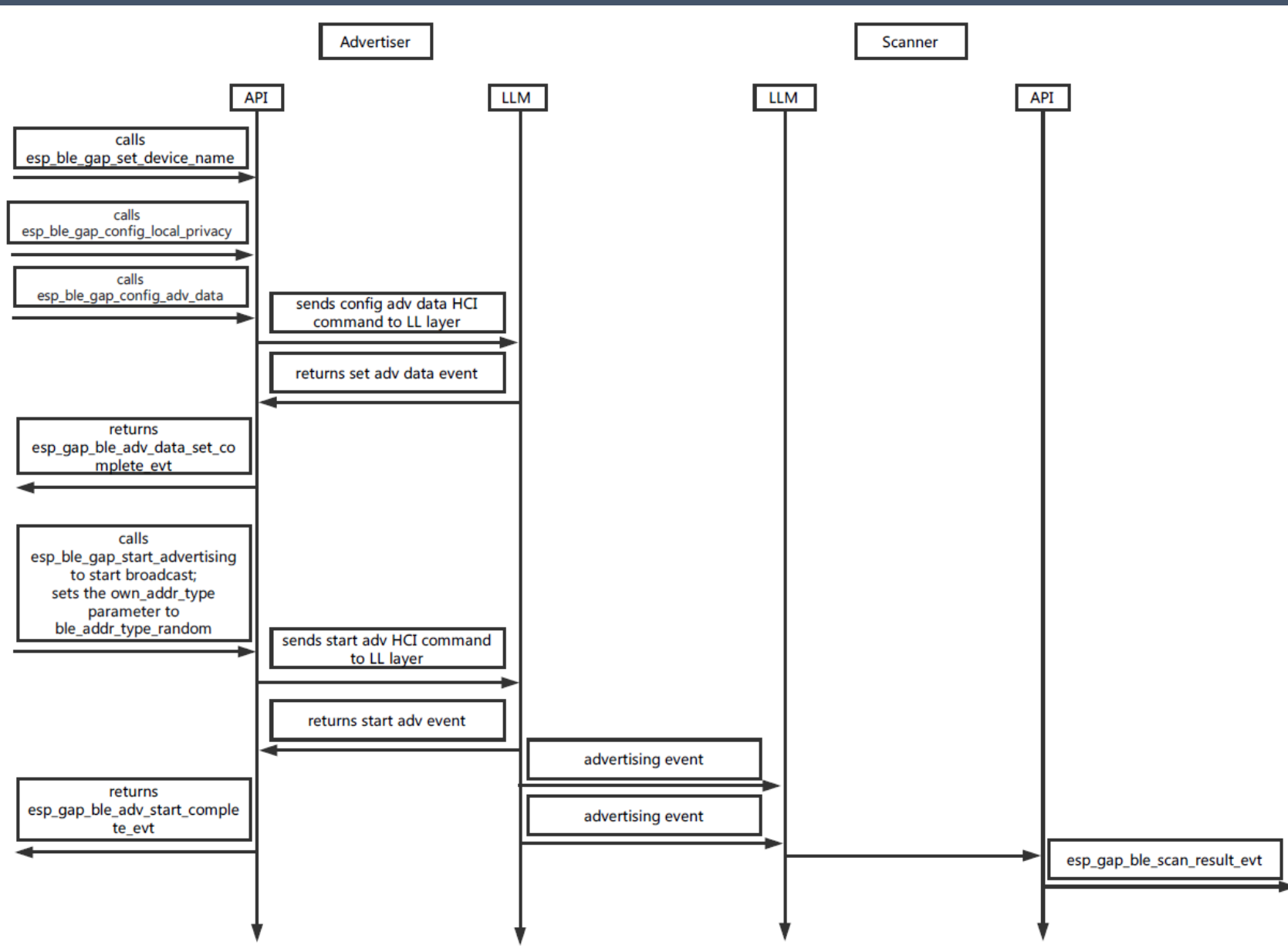
### 1.1.3.2. Broadcast using a resolvable address

resolvable(결정가능한) 주소가 브로드 캐스트에 사용되는 경우 기본 프로토콜 스택은 15 분마다 브로드 캐스트 주소를 업데이트 하고 esp\_ble\_adv\_params\_t의 own\_addr\_type을 **BLE\_ADDR\_TYPE\_RANDOM**으로 설정해야합니다.



### 1.1.3.2. Broadcast using a resolvable address

resolvable(결정가능한) 주소가 브로드 캐스트에 사용되는 경우 기본 프로토콜 스택은 15 분마다 브로드 캐스트 주소를 업데이트 하고 esp\_ble\_adv\_params\_t의 own\_addr\_type을 **BLE\_ADDR\_TYPE\_RANDOM**으로 설정해야합니다.

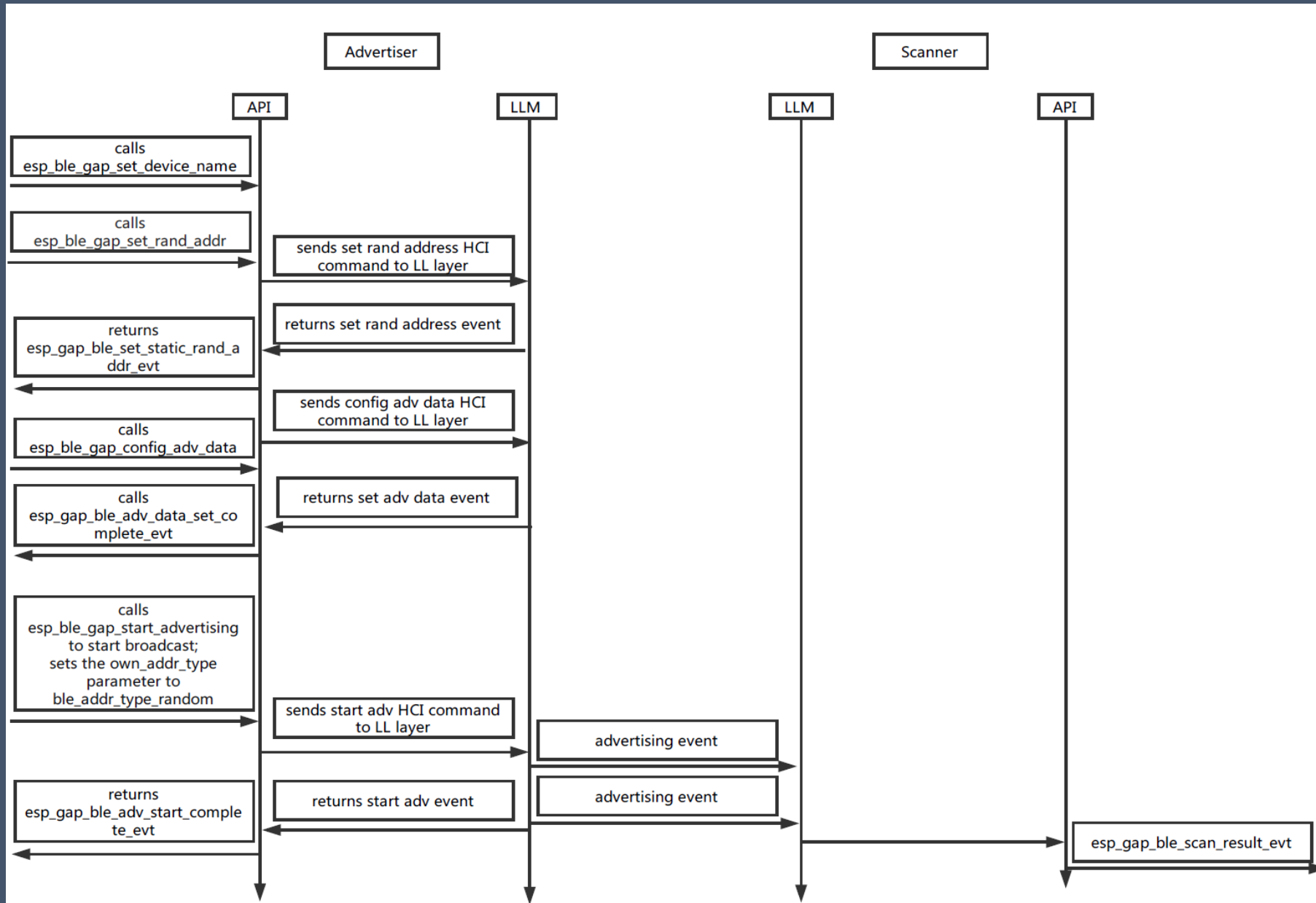


노트 :  
브로드 캐스트에 resolvable한 주소가 사용되는 경우,

**esp\_ble\_gap\_config\_local\_privacy**  
이벤트가 리턴 된 후에만 브로드 캐스트가 시작되고 브로드 캐스트 매개 변수인 own\_addr\_type 유형을 BLE\_ADDR\_TYPE\_RANDOM으로 설정해야합니다.



1.1.3.3. Broadcast using a static random address  
고정 임의 주소가 브로드 캐스트에 사용되는 경우 esp\_ble\_adv\_params\_t의 own\_addr\_type을 BLE\_ADDR\_TYPE\_RANDOM으로 설정해야합니다. 이는 resolvable 주소를 사용하여 브로드 캐스트하는 경우와 유사.





## 1.1.4. BLE Modes

BLE 브로드 캐스트에는 5 개의 모드가 정의되어 있습니다 :

1. 연결 가능 스캔 가능 Undirected Mode
2. 높은 듀티 사이클 Directed mode
3. 스캔 가능 비지정 모드
4. 비 연결 가능 비 지령 모드
5. 연결 가능 낮은 듀티 사이클 Directed mode

#### 1.1.4.1. Connectable Scannable Undirected Mode

**Table 3-1.Packet structure**

Payload	
AdvA (6 octets)	AdvData (0~31 octets)

Connectable Scannable Undirected 모드의 장치는 모든 장치에서 검색하고 연결할 수 있습니다. 스캔 가능성은 피어 장치가 스캔 요청을 보낼 때 로컬 장치가 스캔 응답으로 응답해야 함을 나타냅니다.

위 표와 같이 Connectable Scannable Undirected 브로드 캐스트 패킷은 **6 바이트의 브로드 캐스트 주소와 0 ~ 31 바이트의 브로드 캐스트 패킷 데이터를 포함합니다.** 고정 랜덤 주소가 브로드 캐스트에 사용되는 경우 브로드 캐스트 주소는 esp\_ble\_gap\_set\_rand\_addr을 호출하여 지정됩니다.

공개 주소 또는 resolvable 주소가 브로드 캐스트에 사용되면 브로드 캐스트 주소는 프로토콜 스택에 의해 자동으로 생성됩니다.

#### 1.1.4.2. High Duty Cycle Directed Mode and Connectable Low Duty Cycle Directed Mode

**Table 3-2. Packet structure**

Payload	
AdvA (6 octets)	InitA (6 octets)

IP 지정 브로드 캐스트는 지정된 장치에서만 검색하고 연결할 수 있습니다.

위의 표에 표시된 것처럼 High Duty Cycle Directed Broadcast 패킷에는 6 바이트의 broadcast 장치 주소와 6 바이트의 수신 장치 주소가 포함됩니다. 이 모드에서는 브로드 캐스트 매개 변수 adv\_int\_min 및 adv\_int\_max가 무시됩니다. Connectable Low Duty Cycle Directed 모드에서 브로드 캐스트 매개 변수 adv\_int\_min 및 adv\_int\_max는 **100ms (0xA0)보다 커야합니다.**

Note:

IP directed broadcasts do not carry Adv Data.

### 1.1.4.3. Scannable Undirected Mode

**Table 3-3.Packet structure**

#### **Payload**

AdvA (6 octets)

AdvData (0~31 octets)

Scannable Undirected 모드에서 다른 장치는 장치를 검색 할 수 있지만 연결할 수는 없습니다.

위의 표와 같이 Scannable Undirected 패킷은 6 바이트의 브로드 캐스트 주소와 0 ~ 31 바이트의 브로드 캐스트 패킷 데이터를 포함하며, 이는 Connectable Scannable Undirected 패킷과 동일한 구조입니다.  
이 모드의 장치는 모든 장치에서만 검색 할 수 있지만 연결할 수는 없습니다.

#### 1.1.4.4. Non-connectable Undirected Mode

**Table 3-4.Packet structure**

Payload	
AdvA (6 octets)	AdvData (0~31 octets)

연결 불가능한 무 방향 모드에서는 장치를 장치에 의해 검색 할 수 있지만 다른 장치에 의해 검색되거나 다른 장치에 연결될 수 없습니다.  
스캔 할 수없는 장치는 피어 장치가 스캔 요청을 보낼 때 스캔 응답으로 응답하지 않는 장치입니다.  
연결 해제 가능한 장치는 어떤 장치에도 연결할 수 없는 장치입니다.

위의 표에서 볼 수 있듯이 비 연결 가능 무 지정 브로드 캐스트 패킷에는 6 바이트의 브로드 캐스트 주소와 0 ~ 31 바이트의 브로드 캐스트 패킷 데이터도 포함됩니다.

이 모드에서는 장치를 검색 할 수 있지만 다른 장치로 검색하거나 연결할 수 없습니다.

## 1.1.5. BLE Broadcast Filtering Policy

ESP32의 BLE 아키텍처에서 브로드 캐스트 필터링 정책은 다음 4 가지 값을 가진 `adv_filter_policy` 열거 유형을 설정하여 구현됩니다.

- `ADV_FILTER_ALLOW_SCAN_ANY_CON_ANY`
- `ADV_FILTER_ALLOW_SCAN_WLST_CON_ANY`
- `ADV_FILTER_ALLOW_SCAN_ANY_CON_WLST`
- `ADV_FILTER_ALLOW_SCAN_WLST_CON_WLST`

네 가지 값은 각각 4 가지 경우에 해당합니다.

- 스캔하고 모든 장치에 연결할 수 있습니다 (화이트리스트 없음)
- 모든 연결 요청과 화이트리스트의 검색 요청 만 처리합니다.
- 모든 스캔 요청과 화이트리스트의 연결 요청 만 처리합니다.
- 화이트리스트에서 연결 요청 및 스캔 요청 처리

## 1.1.6. BLE Scanning Procedure

ESP32에서 스캐닝 장치는 주로 `esp_ble_gap_set_scan_params`를 호출하여 스캔 매개 변수를 설정 한 다음 호출하여 스캔을 시작합니다.

`esp_ble_gap_start_scanning`. 스캔한 장치의 정보는 `ESP_GAP_BLE_SCAN_RESULT_EVT` 이벤트 또는 기간이 시간 초과 될 때 `ESP_GAP_SEARCH_INQ_CMPL_EVT` 이벤트에 의해 반환됩니다.



Notice:

When the value of the duration is 0, the device will be scanned permanently without timeout.

지속 시간 값이 0이면 시간 초과 없이 장치가 영구적으로 검색됩니다. ( `SCAN_ALL_THE_TIME` )



## 1.1.6. BLE Scanning Procedure

ESP32에서 스캐닝 장치는 주로 `esp_ble_gap_set_scan_params`를 호출하여 스캔 매개 변수를 설정 한 다음 호출하여 스캔을 시작합니다.

`esp_ble_gap_start_scanning`. 스캔한 장치의 정보는 `ESP_GAP_BLE_SCAN_RESULT_EVT` 이벤트 또는 기간이 시간 초과 될 때 `ESP_GAP_SEARCH_INQ_CMPL_EVT` 이벤트에 의해 반환됩니다.



Notice:

When the value of the duration is 0, the device will be scanned permanently without timeout.

지속 시간 값이 0이면 시간 초과 없이 장치가 영구적으로 검색됩니다. ( `SCAN_ALL_THE_TIME` )

## 1.1.7. BLE GAP Implementation Mechanism

ESP32는 BLE GAP API를 호출하고 BLE GAP 콜백을 등록하며 반환 된 이벤트 값으로 현재 장치의 상태를 얻습니다.

# 1.2. GATT

## 1.2.1 ATT

BLE 아키텍처 내부의 데이터는 다음 네 가지 기본 요소로 구성되는 속성 형식으로 존재합니다.

- 속성 핸들 : 속성 핸들은 주소를 사용하여 메모리에서 데이터를 찾는 것과 유사한 속성을 찾는 데 도움이 됩니다.  
(Handle) 예를 들어, 첫 번째 속성의 핸들은 0x0001이고 두 번째 속성은 0x0002 등이며 최대 0xFFFF입니다.
- 속성 유형 : 각 데이터 세트는 온도, 전송 전력, 배터리 잔량 등과 같은 특정 유형의 정보를 노출합니다.  
(Type) 노출되는 데이터의 유형을 속성 유형이라고 합니다. 노출 될 수 있는 서로 다른 가능한 유형의 데이터가 주어지 UUID라고도하는 16 비트 또는 128 비트 숫자가 속성의 유형을 식별하는 데 사용됩니다.  
예를 들어 UUID 0x2A09는 배터리 수준, UUID 0x2A6E는 온도입니다.
- 속성 값 : 속성 값은 각 속성의 핵심 정보이며 다른 세 가지 요소 (핸들, 유형 및 권한)가 추가되어 속성 값을 더 잘 이해할 수 있습니다.  
(value) 다른 속성 유형에 대한 속성 값의 길이는 고정되거나 가변적 일 수 있습니다. 예를 들어 배터리 수준의 속성 값 길이 BLE 사용 패스 스루 모듈의 속성 길이는 가변적 인 반면 배터리 수준 속성의 가능한 모든 값 (예 : 0-100)을 포함하기에 충분한 1 바이트입니다.
- 속성 권한 : 각 속성에는 읽거나 쓸 수 있는 정보가 포함됩니다.  
(Permission) 액세스시 이러한 제한을 용이하게 하기 위해 각 속성마다 고유 한 속성 권한이 있습니다.  
데이터를 소유 한 당사자는 속성 권한을 통해 로컬 데이터의 읽기 / 쓰기 액세스를 제어 할 수 있습니다.

## 1.2. GATT

### 1.2.1 ATT

Table 3-5. Attribute Structure Table

Attribute Handle	Attribute Type	Attribute Value	Attribute Permission
0x0001	-	-	-
0x0002	-	-	-
Attribute Handle	Attribute Type	Attribute Value	Attribute Permission
.....	-	-	-
0xFFFE	-	-	-
0xFFFF	-	-	-

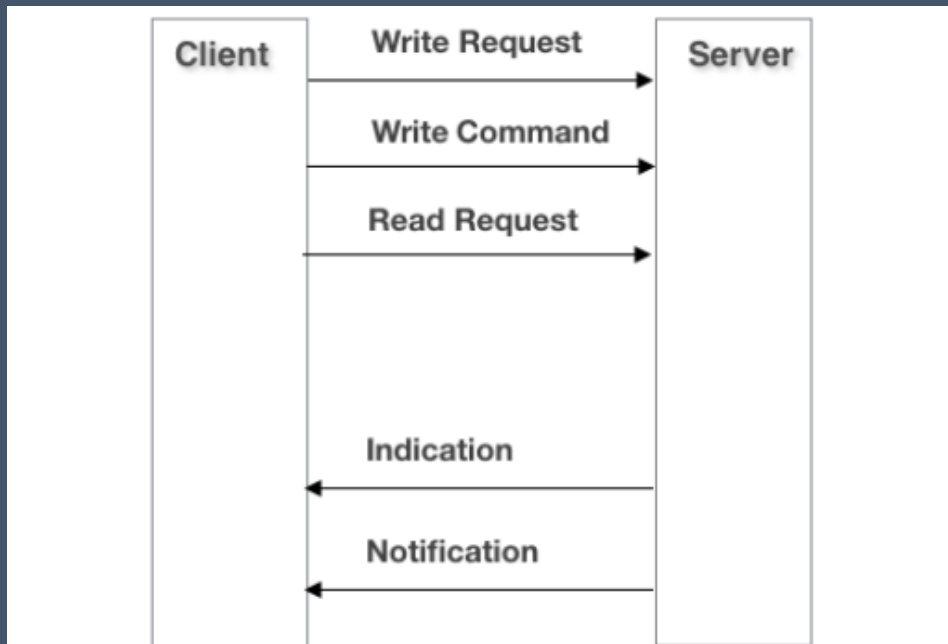
데이터를 보유한 장치 (즉, 속성)는 서버로 정의되고 서버에서 데이터를 얻는 장치는 클라이언트로 정의됩니다.

## 1.2. GATT

### 1.2.1 ATT

서버와 클라이언트 간의 일반적인 작업은 다음과 같습니다.

- 클라이언트는 서버에 데이터를 기록하여 서버로 데이터를 보냅니다. 쓰기 요청 및 쓰기 명령 모두 속성 값을 쓰는 데 사용될 수 있습니다. 그러나 쓰기 요청은 쓰기 요청이 사용될 때만 프롬프트됩니다.
- 서버는 클라이언트에게 표시 또는 알림을 보내 데이터를 클라이언트로 보냅니다. 표시와 알림의 유일한 차이점은 표시가 사용될 때만 확인 메시지가 표시된다는 것입니다. 이것은 쓰기 요청과 쓰기 명령의 차이점과 유사합니다.
- 클라이언트는 읽기 요청을 시작하여 서버에서 데이터를 얻을 수도 있습니다.



표시 (Indication), (\*) IND (indication)  
알림 (Notification)

Figure 3-5. Common operations between a server and a client

## 1.2. GATT

### 1.2.1 ATT

검색결과  
**MTU**란 TCP/IP 네트워크 등과 같은 패킷 또는 프레임 기반의 네트워크에서 전송될 수 있는 최대 크기의 패킷 또는 프레임을 말한다. 한번에 전송할 수 있는 최대 전송량(Byte)인 **MTU** 값은 매체에 따라 달라진다

주의:

서버와 클라이언트 간의 일반적인 작업에 대한 자세한 내용은 Core\_V5.0, Vol3을 참조하십시오. F부, 3.4 장 “속성 프로토콜 PDU”.

서버와 클라이언트 간의 공통 작업은 ATT PDU를 사용하여 수행됩니다. 각 장치는 지원되는 **MTU**를 지정할 수 있으며 이는 ATT 메시지의 최대 길이입니다. ESP32 IDF에서 MTU는 23-517 바이트 일 수 있으며 속성 값의 총 길이는 제한되지 않습니다.

사용자의 패킷 길이가 (MTU-3)보다 길면 데이터를 쓰려면 쓰기 요청 준비가 필요합니다. 유사하게, 패킷의 길이가 (MTU-1)보다 길면, 나머지 데이터를 판독하기 위해 판독 Blob 요청이 요구된다.

⚠ \*주의 사항 :

단일 물리적 패킷의 MTU와 LE 패킷 길이의 차이점이 여기에 강조 표시되어야 합니다. 이 경우 호스트 ATT 계층에 관한 MTU는 모든 "전송할 데이터"가 단일 ATT 요청에 적합 할 수 있는지 여부와 데이터를 전송하기 위해 준비 쓰기 요청이 필요한지 여부를 결정합니다. 한편, PHY 계층에 관한 LE 패킷 길이는 패킷이 여러 패킷으로 분할되고 전송 될 필요가 있는지를 결정한다. 예를 들어, (MTU + 4)가 LE 패킷 길이보다 크면이 ATT 패킷을 두 개 이상의 물리적 패킷으로 나누어 전송해야 합니다. 반대로, (MTU + 4)가 LE 패킷 길이보다 짧은 경우 전체 ATT 패킷을 전송하는 데 하나의 물리적 패킷 만 필요합니다. MTU에 4 바이트를 추가하는 이유는 전송 중에 4 바이트 L2CAP 헤더가 추가되기 때문입니다.

## 1.2. GATT

### 1.2.2 GATT Profile

www.CodeZoo.co.kr

ATT는 BLE 아키텍처에서 최소 데이터 저장 단위를 지정하고, GATT는 속성 값 및 설명자를 사용하여 데이터 세트를 표시하는 방법, 유사한 데이터를 서비스로 집계하는 방법 및 피어 장치가 어떤 서비스 및 데이터를 찾는 지 정의합니다. GATT는 아래에 요약 된 경우와 같이 순수한 숫자가 아닌 정보에 관한 특성 개념을 소개합니다.

- 주어진 값의 단위 (예 : 킬로그램 (kg)로 측정 된 무게, 섭씨 (°C)로 측정 된 온도 등).
- 주어진 값의 이름. 예를 들어 동일한 UUID를 가진 특성의 경우 (예 : 온도 속성에서 값의 이름은 피어 장치에이 값이 "마스터 침실의 온도"를 나타내고 다른 하나는 "거실의 온도"를 나타냅니다).
- 230,000 및 460,000과 같은 과도한 데이터 수의 지수. 지수가 **이미  $10^4$ 로 지정된 경우, "23"과 "46"만 전송하면 230,000과 460,000을 나타내는 데 충분합니다.** 실제 응용 프로그램에서 데이터를 정확하게 설명하기위한 기존 요구 사항 중 몇 가지 예입니다. 보다 미묘한 정보를 제공하려면이 추가 정보를 각각에 저장하기 위해 큰 데이터 공간을 예약해야 합니다. 특성. 그러나 대부분의 경우 예약 된 추가 공간은 대부분 사용되지 않습니다. 따라서 이러한 설계는 가능한 프로토콜만큼 간결해야 하는 BLE의 전제 조건을 준수하지 않습니다. 이와 같은 경우, GATT 스펙은 이 추가 정보를 간략하게 설명하기 위해 디스크립터 개념을 도입합니다. 각 데이터 조각과 디스크립터에는 일대일 대응이 없습니다. 즉, 복잡한 데이터는 간단한 데이터에는 디스크립터가 없을 수 있습니다. 특성은 세 가지 기본 요소로 구성됩니다.
- **특성 선언** : 선언은 특성의 시작으로, 선언 후 콘텐츠가 특성 값을 피어 장치에 알립니다. 두 선언 사이의 모든 핸들은 완전한 특성을 구성합니다. 쓰기 읽기 속성도 선언에 포함됩니다.
- **특성 값** : 특성 값은 특성의 주요 부분으로 특성의 가장 중요한 정보를 전달합니다.
- **설명자** : 설명자는 특성 (예 : 구성 정보 제공)을 추가로 설명 할 수 있으며 특성에는 설명자가 여러 개이거나 없을 수 있습니다. BLE에서 GAP는 유사한 기능을 서비스 형태로 그룹화합니다. 예를 들어, 배터리와 관련된 모든 특성과 동작은 배터리 서비스로 정의 할 수 있습니다. 심박수와 관련된 모든 특성과 행동은 심박수 서비스로 정의 될 수 있습니다. 체중계와 관련된 모든 특성과 행동 체중계 서비스로 정의 될 수 있습니다. 서비스는 일반적으로 하나 이상의 특성을 포함하며 각 특성은 0 개 또는 많은 설명자를 포함합니다. 사용자는 자신의 응용 프로그램 요구 사항에 따라 필요한 서비스를 선택하고 최종 응용 프로그램을 구성 할 수 있습니다.

# 1.2. GATT

## 1.2.2 GATT Profile

\*Notices:

- For other definitions of services, characteristics and descriptors, please refer to:
- Chapter 3 "Service Interoperability Requirements", Part G, Vol3., Core\_V5.0;
  - <https://www.bluetooth.com/zh-cn/specifications/gatt>

Table 3-6. The definition table of services

Attribute Handle	Attribute Type
0x0001	Service 1
0x0002	Characteristic Declaration 1
0x0003	Characteristic Value 1
0x0004	Descriptor 1
0x0005	Characteristic Declaration 2
0x0006	Characteristic Value 2
0x0007	Descriptor 2
0x0008	Descriptor 3
0x0009	Service 2
.....	.....

완성된 서비스 테이블



## 1.2. GATT

### 1.2.3 Add Gatt Services in ESP32 IDF Environment

사용자는 이벤트를 통해 ESP32 IDF 릴리스 1.0에서 하나씩 서비스와 특성을 수동으로 추가 할 수 있습니다. 모든 읽기-쓰기 작업은 이벤트를 통해 응용 프로그램 계층에 도달하며 사용자는 패키지를 사용하여 응답 할 수 있습니다. 이 접근 방식은 특히 대형 GATT 데이터베이스에 서비스를 추가 할 때 BLE 프로토콜에 익숙하지 않은 사용자의 오류가 발생하기 쉽습니다. 따라서 서비스를 하나씩 수동으로 추가하지 않는 것이 좋습니다.

#### \*Notice:

The interface and examples of adding services and characteristics manually are still reserved in ESP IDF. For more information, please refer to the gatt\_service example.

이 컨텍스트에서 속성 테이블을 사용하여 서비스 및 특성을 추가하는 것이 ESP32 IDF 릴리스 2.0에 도입되었습니다. 사용자는 속성 테이블에 새 서비스 및 특성을 입력 한 다음 `esp_ble_gatts_create_attr_tab`을 호출하여 새 서비스 및 특성을 추가 할 수 있습니다. 또한 이 경우 하위 계층 응답도 지원되므로 하위 계층은 일부 요청에 응답하고 오류를 자동으로 식별 할 수 있으므로 사용자는 데이터 수신 및 전송에 집중할 수 있습니다. 이러한 방식으로 사용자는 BLE 사양을 다시 구현할 필요없이 다른 플랫폼에서 쉽게 ESP32 플랫폼으로 프로파일을 포팅 할 수 있습니다.

#### \*Notice:

We recommend that users add services and characteristics **through the attribute table**, which is much easier, less error-prone, and supports low-layer responses. For details, please refer to the gatt\_server\_service\_table examples.

속성 테이블의 구조는 `esp_gatts_attr_db_t`를 통해 속성을 설명하기 위해 초기화가 필요한 모든 매개 변수를 정의합니다.

Table 3-7. The structure parameters of ESP32 IDF

Parameter	Description
<code>uint8_t attr_control</code>	주어진 <code>write_response</code> 와 같은 일부 응답을 정의합니다. 하위 계층에서 자동으로 또는 애플리케이션 계층으로 전달되므로 사용자가 수동으로 응답 할 수 있습니다. <b>ESP_GATT_AUTO_RSP</b> 자동 응답 모드가 권장됩니다.
<code>uint16_t uuid_length</code>	UUID의 길이가 16 비트, 32 비트 또는 128 비트임을 나타냅니다. 속성 UUID는 포인터에 의해 전송되므로 UUID의 길이를 지정해야 합니다.

Parameter	Description
<code>uint8_t *uuid_p</code>	<p>현재 속성의 UUID 포인터를 나타냅니다. 사용자는 <code>uuid_lenght</code> 매개 변수에 지정된 길이 정보를 기반으로 특정 길이의 UUID 값을 읽을 수 있습니다.</p> <p>-----</p>
<code>uint16_t perm</code>	<p>현재 속성의 쓰기 및 읽기 권한을 나타냅니다. 이 매개 변수는 비트 단위로 작동합니다. 각 비트는 특정 쓰기 및 읽기 권한을 나타냅니다. 특정 비트를 조작하면 해당 속성의 쓰기 및 읽기 권한이 변경 될 수 있습니다. 예를 들어, <code>PERM_READ   PERM_WRITE</code>는 읽고 쓸 수있는 속성을 의미합니다.</p> <p>-----</p>
<code>uint16_t max_length</code>	<p>현재 속성 값의 최대 길이를 나타냅니다. 프로토콜 스택은이 매개 변수를 기반으로 속성에 메모리를 할당합니다. 피어 장치가 기록한 속성 값의 길이가이 매개 변수에 정의 된 최대 길이를 초과하면 쓰기 오류가 발생하여 쓰기 작업의 길이가 데이터의 최대 길이를 초과 함을 나타냅니다.</p> <p>-----</p>
<code>uint16_t length</code>	<p>현재 속성의 실제 길이를 나타냅니다. 예를 들어, 속성의 최대 길이가 512 비트이고 피어 장치가이 속성에 "0x1122"를 쓰려는 경우 속성의 현재 길이를 2로 설정합니다. 그러면 피어 장치가이 속성을 읽습니다. 메모리에서이 속성의 실제 길이를 얻을 수 있으며 전체 512 비트 대신 실제 값이있는 부분 만 보낼 수 있습니다.</p> <p>-----</p>
<code>uint8_t *value</code>	<p>현재 속성 값의 초기화 된 값을 나타냅니다. 이후 이 매개 변수의 형식은 포인터이며,이 매개 변수의 실제 길이는 포인터에서 정확한 값을 얻기 위해 <code>length</code> 매개 변수에 의해 먼저 얻어져야합니다.</p>

## 1.2. GATT

### 1.2.4 Discover a Peer Device's Services in ESP32 IDF (GATT Client)

검색 서비스는 **GATT 클라이언트**가 피어 장치의 서비스 및 특성을 검색하는 데 도움이 될 수 있습니다. 검색 절차는 장치마다 다를 수 있습니다. 피어 장치의 GATT 서비스를 검색하는 방법의 예와 함께 ESP32 IDF의 검색 절차가 여기에 소개됩니다.

- 먼저, 서비스를 포함한 모든 피어 장치의 서비스 정보를 발견하십시오

UUID 및 속성 핸들 범위.

- GATT 서비스, UUID 0x1801, 0x0001 ~ 0x0005 처리
- GAP 서비스, UUID 0x1800, 0x0014 ~ 0x001C 처리
- 그런 다음 핸들 범위 내에서 모든 피어 장치의 특성을 발견하십시오.


GATT 서비스 (0x0001 ~ 0x0005).

- 서비스 변화 특성 찾기, 핸들 0x0002 ~ 0x0003
- 0x0002는 특성 선언을 나타냅니다.
- 0x0003은 특성 값을 나타냅니다.

따라서 각 특성에는 적어도 두 개의 핸들 속성이 있습니다.

- GATT 서비스의 핸들이 0x0001 ~ 0x0005, 0x0003 범위에있는 경우 해당 설명자가 뒤에 와야합니다. 따라서 모든 설명자는 0x0004부터 찾아야합니다.

- 0x0004는 클라이언트 특성 구성의 설명자를 나타냅니다.
- 0x0005에는 현재 정보가 없으며 핸들이 될 수 있습니다  
이 서비스를 위해 예약되었습니다.
- 이 시점에서 GATT 서비스의 검색 절차가 완료되었습니다.

```
esp_err_t esp_ble_gap_config_adv_data_raw(uint8_t *raw_data, uint32_t raw_data_len) 
```

This function is called to set raw advertising data. User need to fill ADV data by self.

#### Return

- ESP\_OK : success
- other : failed

#### Parameters

- `[in] raw_data` :: raw advertising data
- `[in] raw_data_len` :: raw advertising data length , less than 31 bytes

```
esp_err_t esp_ble_gap_config_scan_rsp_data_raw(uint8_t *raw_data, uint32_t raw_data_len)
```

This function is called to set raw scan response data. User need to fill scan response data by self.

#### Return

- ESP\_OK : success
- other : failed

#### Parameters

- `[in] raw_data` :: raw scan response data
- `[in] raw_data_len` :: raw scan response data length , less than 31 bytes

[ ESP\_GATTS\_DEMO ]  
advertising and scan response data

Advertising Data :

02 01 06 0F 09 45 53 50 5F 47 41 54 54 53 5F 44  
45 4D 4F 05 03 EE 00 FF 00 05 12 06 00 10 00  
02 01 06 0F 09 45 53 50 5F 47 41 54 54 53 5F 44  
45 4D 4F 02 0A 03 05 03 EE 00 FF 00 00 00 00

02 01 06

size : 2(0x02) bytes

data type : 0x01  
(Flags)

data :

06

0 0 0 0 0 1 1 0 b

(0x02)LE General Discoverable Mode

(0x04)BR/EDR Not Supported

0F 09 45 53 50 5F 47 41 54 54 53 5F 44 45 4D 4F

size : 15(0x0F) bytes

data type : 0x09  
(Complete Local Name)

data :

45 53 50 5F 47 41 54 54 53 5F 44 45 4D 4F

device name : ESP\_GATTS\_DEMO

05 03 EE 00 FF 00

size : 5(0x05) bytes

data type : 0x03

(Complete List of 16-bit Service  
Class UUIDs)

data :

EE 00 FF 00

UUID : 0x00EE

05 12 06 00 10 00

size : 5(0x05) bytes

확인

# 1.3. SMP(Security Management Protocol)

## 1.3.1 Overview

SMP 관련 API는 ESP32 BLE의 GAP 모듈에 패키징되어 있습니다.

SMP는 암호화 키와 ID 키를 생성하고 페어링 및 키 분배를 위한 편리한 프로토콜을 정의하며 프로토콜 스택의 다른 계층이 다른 장치와 안전하게 데이터를 연결하고 교환 할 수 있도록합니다. 이 프로세스에서는 데이터 링크 계층의 연결 및 특정 보안 표준이 필요합니다. GAP SMP를 사용하면 Bluetooth Core Specification 버전의 SMP 장과 같은 보안 수준을 설정하여 두 장치가 데이터 링크 계층에서 연결을 암호화 할 수 있습니다.

GAP SMP의 구현하기 앞서 아래 개념을 명확히 알아야 합니다.

- 페어링
- 본딩
- Authentication
- Authorization



- 페어링(Pairing)

두 장치가 특정 보안 수준으로 연결하기로 동의했음을 나타냅니다.

- 본딩(Bonding)

적어도 하나의 장치가 어떤 종류의 표시를 보냈 음을 나타냅니다.

향후 연결을 위해 LTK, CSRK 또는 IRK 일 수있는 보안 정보. 이 두 장치가 서로 결합 할 수있는 경우 페어링 후에 키 분배가 발생합니다. 그렇지 않으면 결합 정보가 교환되지 않습니다. 본딩은 페어링의 전제 조건이 아닙니다. 그러나, 페어링하는 동안, 두 장치는 그들의 특성을 교환하여 피어 장치가 본딩을 위해 개방되어 있는지를 결정한다. 이 두 장치 중 어느 것도 본딩을 위해 열려 있지 않으면 피어 장치의 보안 정보를 저장하지 않아야합니다.

- 인증(Authentication)

링크의 보안을 나타냅니다. 그러나 비인증 링크는

반드시이 링크가 전혀 안전하지 않다는 것을 의미하지는 않습니다. 링크 암호화 키에 두 장치에서 모두 확인한 보안 속성이 있으면이 두 장치는 인증 된 것으로 간주됩니다. STK를 인증에 사용하면 페어링 중에 키워드가 생성됩니다. 입 / 출력 및 OOB 장치

기능, 생성 및 교환 된 모든 키는 MITM 속성을 갖습니다 (보안을 강화하는 PIN / 큰 OOB 키가 사용됨). Just Works를 사용하는 경우 생성 및 교환 된 모든 키에는 No MITM 속성이 있습니다.

- 인증(Authorization)

작업 수행 권한 부여로 정의

응용 프로그램 계층에서. 일부 응용 프로그램은 인증이 필요할 수 있으며,이 경우 응용 프로그램을 사용하기 전에 권한을 부여 받아야합니다. 권한이 없으면 전체 프로세스가 실패합니다.

## 1.3.2 Safety Management Controller

### 1.3.2.1 BLE Encryption

BLE 장치의 암호화는 다음 두 가지 기본 방법으로 수행 할 수 있습니다.

- 두 BLE 장치간에 본딩이 설정되지 않은 경우 이러한 장치는 페어링 절차를 통해 암호화되며 본딩 (또는 본딩 되지 않음)은이 BLE 장치의 특정 페어링 정보에 따라 결정됩니다.
- 두 개의 본딩 장치 : 본딩 절차를 통해 암호화를 시작합니다. 두 장치가 이미 본딩 된 경우 원래 본딩 프로세스에 의존하여 한 장치로 암호화가 시작됩니다.

마스터가 Just Works 모드에서 암호화 요청을 시작하는 방법은 다음 순서도에서 확인할 수 있습니다.

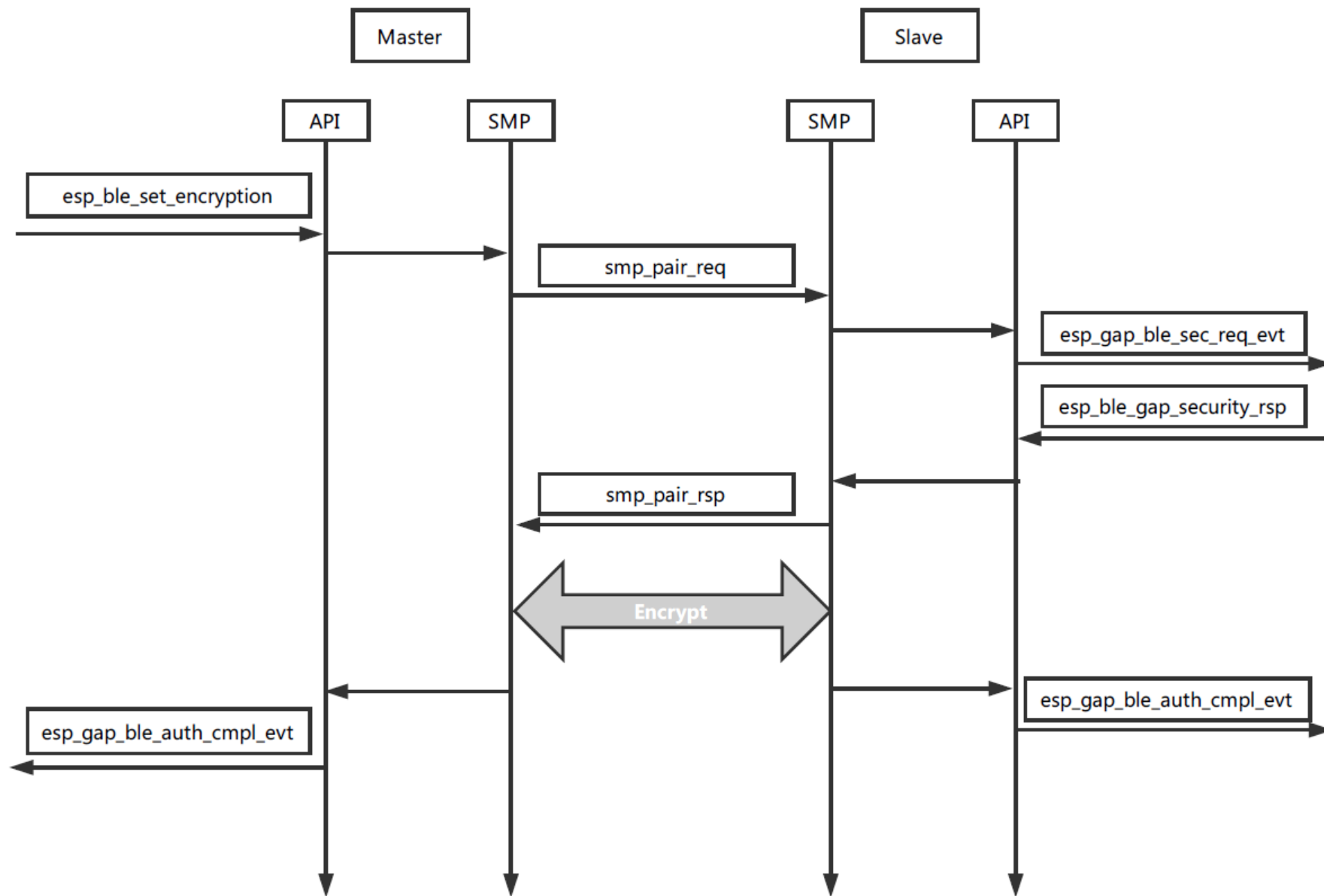


Figure 3-6. The flow chart of encryption in Just Works mode

마스터가 패스 키 알림 모드에서 암호화 요청을 시작하는 방법은 아래 순서도에서 확인할 수 있습니다.

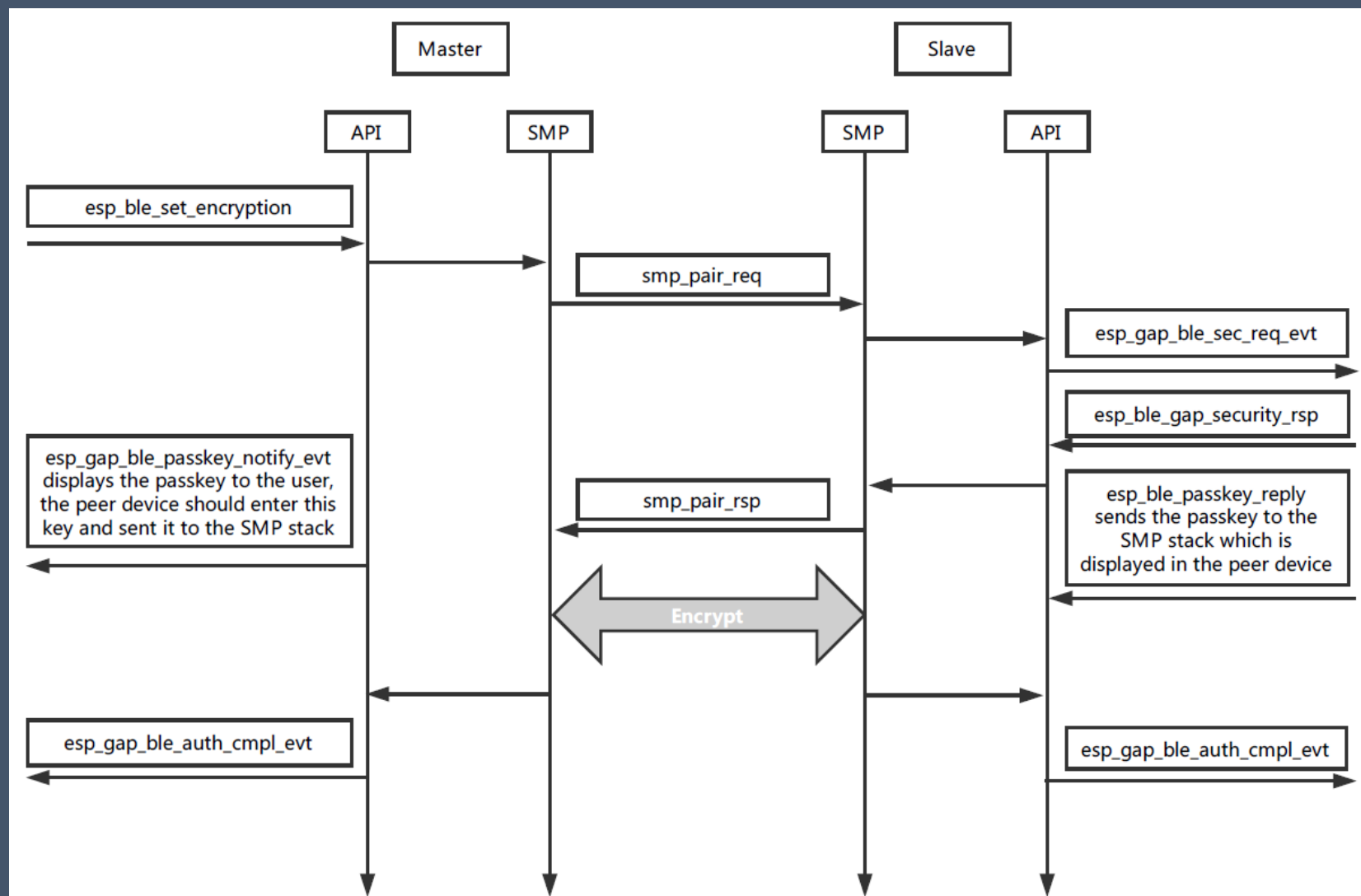


Figure 3-7. The flow chart of encryption in Passkey Notify mode

### 3.3.2.2.BLE Bonding

두 BLE 장치 간의 결합은 GAP API를 호출하여 수행됩니다. Bluetooth 핵심 사양의 설명에 따르면, 본딩의 목적은 **SMP에 의해 암호화 된 두 개의 BLE 장치가 동일한 키를 사용하여 서로 다시 연결할 때 링크를 암호화하여 다시 연결 프로세스를 단순화 할 수 있다는 것입니다.** 이 두 BLE 장치는 페어링 중에 암호화 키를 교환하여 장기간 사용하기 위해 저장합니다. 본딩 프로세스는 아래 순서도에서 확인할 수 있습니다.

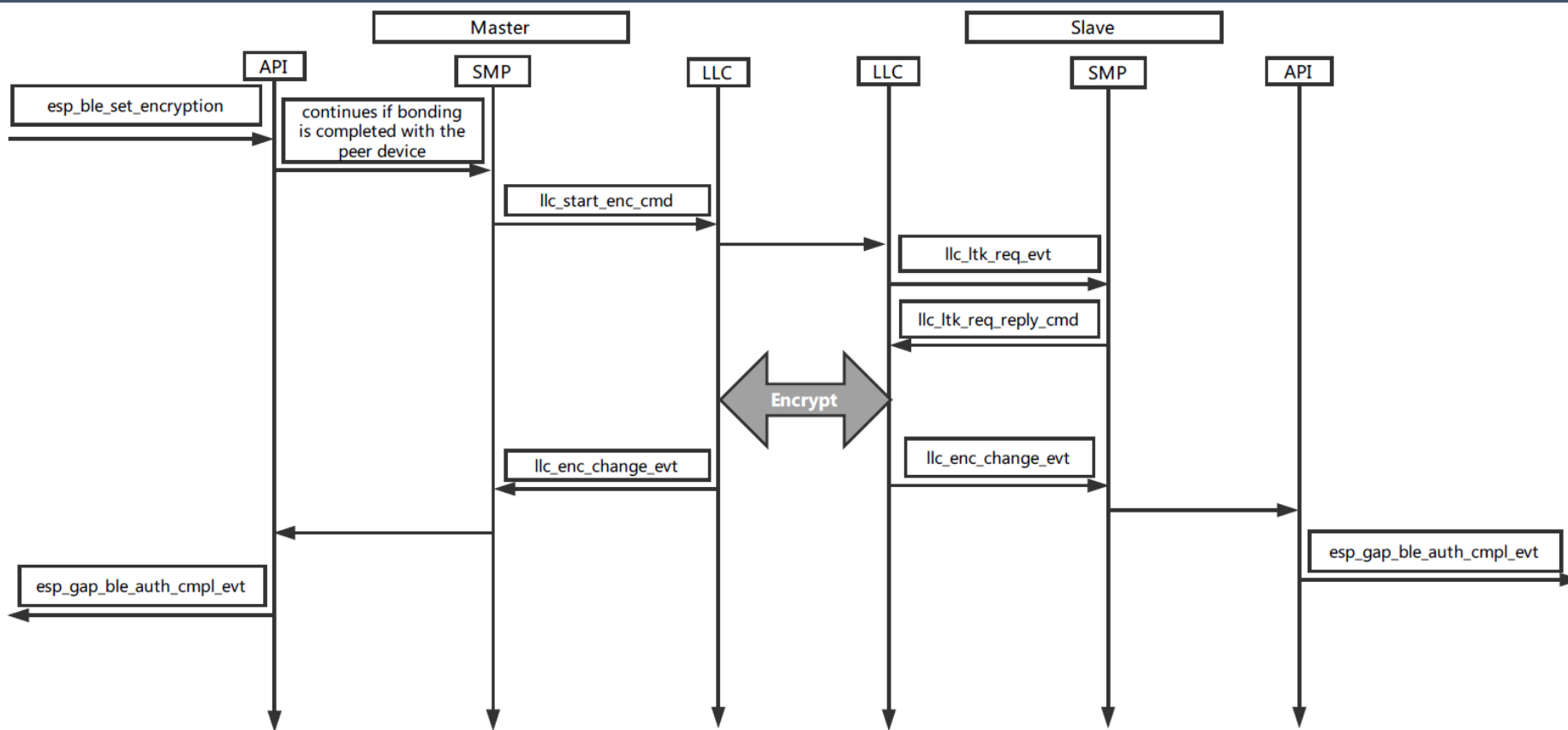


Figure 3-8. The flow chart of BLE bonding process

주의:  
연결하는 동안 마스터 장치가 본딩 프로세스를 시작해야 합니다.

### 1.3.3. The Implementation of SMP

BLE SMP는 BLE GAP에서 암호화 API를 호출하고 BLE GAP 콜백을 등록하며 이벤트의 반환 값을 통해 현재 암호화 상태를 얻습니다.

## 2. EspBlufi

단일 2.4GHz Wi-Fi 및 Bluetooth 콤보 칩인 ESP32는 **SmartConfig** 및 **Bluetooth**를 통해 **Wi-Fi 설정을 지원**합니다. 사용자는 ESP32를 사용하여 IoT 장치의 Wi-Fi 네트워킹을 안전하게 구성 할 수 있습니다.

Wi-Fi 네트워크 구성에 Bluetooth를 사용하면 다음과 같은 장점이 있습니다.

- Bluetooth 프로토콜이 열려 있고 확장 가능합니다.
- 블루투스 프로토콜을 사용하면 블루투스 비콘을 통해 주변 장치를 쉽게 찾을 수 있습니다.
- 암호가 장치로 전송되기 전에 장치의 인증이 보안 Bluetooth 연결을 통해 수행되므로 Bluetooth 프로토콜이 안전합니다.
- 라우터가 작동하지 않더라도 Bluetooth를 통해 스마트 폰으로 데이터를 전송할 수도 있습니다. 전화기는 데이터를 인터넷에 업로드 할 수 있습니다.
- 이제 Wi-Fi 네트워크가 다운 된 경우 전화기에서 Bluetooth 장치에 연결하고 장치를 제어하기위한 명령을 직접 보낼 수도 있습니다.

### 1.2 EspBlufi

ESP32는 Bluetooth v4.2 BR / EDR 및 BLE 사양을 준수합니다.

Espressif는 IoT 장치의 Bluetooth 네트워킹을 위해 특별히 EspBlufi 앱을 개발했습니다.

이 앱은 Android 4.3 이상에서 지원됩니다. Android 6.0 이상의 스마트 폰의 경우 Google이 Android API를 수정했기 때문에 사용자는 현재 위치에 액세스 할 수있는 권한을 부여하고 위치 정보 모듈이 Bluetooth 스캔을 시작하도록해야 합니다.



## 2. EspBlufi

- EspBlufi can be downloaded via: [https://github.com/EspressifApp/EspBlufiForAndroid/ releases](https://github.com/EspressifApp/EspBlufiForAndroid/releases)

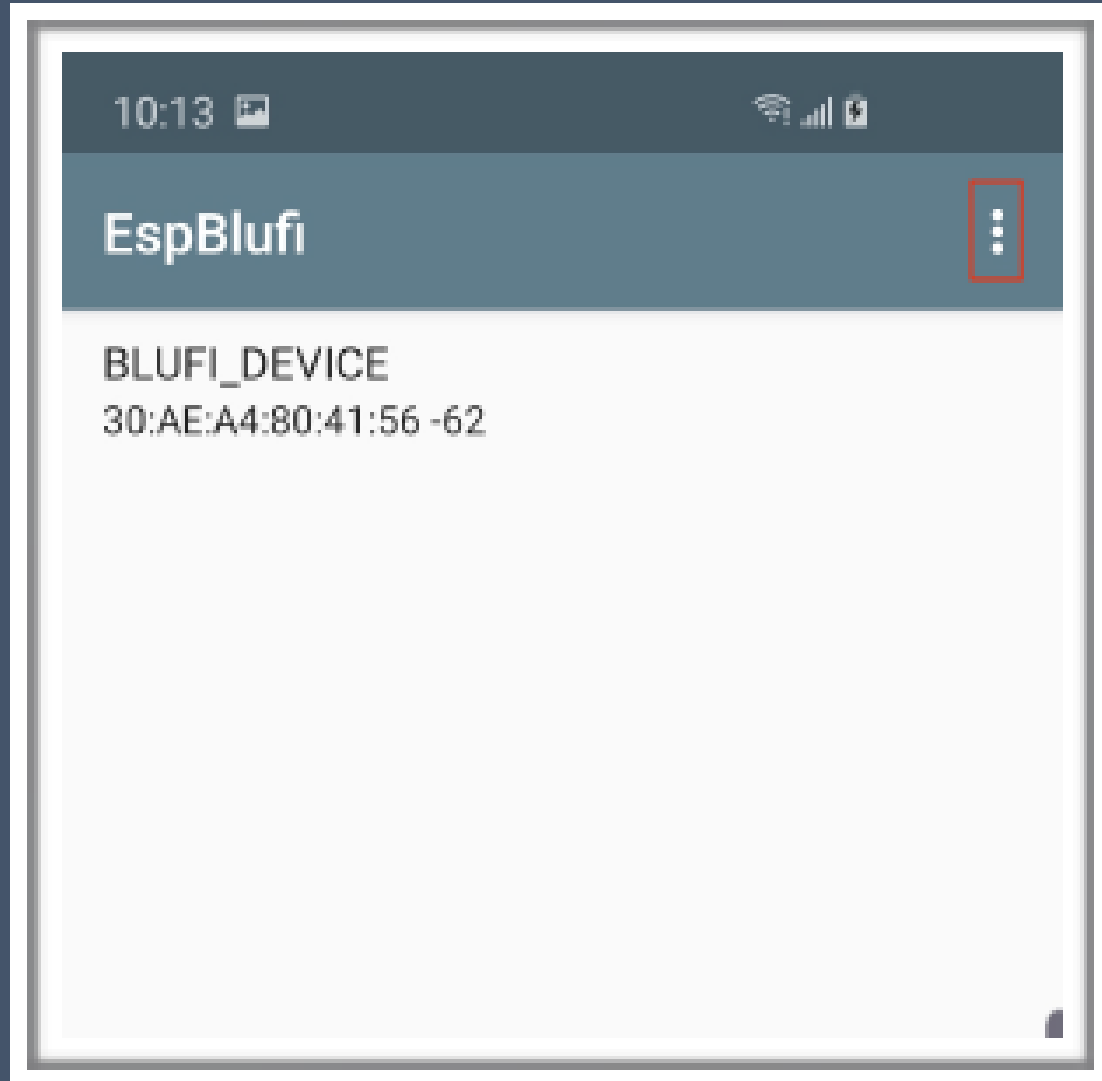
<source code>

<https://github.com/EspressifApp/EspBlufiForAndroid>

<https://github.com/EspressifApp/EspBlufiForiOS>

## 2. EspBlufi

- Open EspBlufi after downloading it. In the user interface as shown in Figure 1-1, click on the `:` icon in the upper right corner.



## 2. EspBlufi

- The Configure button will be shown in the interface and click this button.



Figure 1-2. CONFIGURE Button

## 2. EspBlufi

- Configure the length of BLE mtu, BLE device filtering, and check the version of the app and Blufi repository.



Figure 1-3. CONFIGURE Interface

## 2. EspBlufi

### 2.1. BluFi Protocol

Protocol documentation: <https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/blufi.html>

*감사합니다.*